NATURAL LANGUAGE EVALUATION WITH HUMANS IN THE LOOP AND STATISTICAL ESTIMATORS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Arun Tejasvi Chaganty

August 2018

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Percy S. Liang) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Christopher D. Manning)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Michael S. Bernstein)

Approved for the Stanford University Committee on Graduate Studies

Preface

In natural language tasks such as knowledge base population, text summarization or open-response question answering, a significant challenge is simply evaluating the performance of automated systems because of the large diversity of possible outputs. Existing fully-automatic methods for evaluating these systems rely on an *incomplete* set of annotated references which lead to *systematic biases* against certain system improvements: in other words, genuinely good ideas are systematically discarded simply because of limitations in our evaluation methodology. As a result, human evaluation, which can be prohibitively expensive, has remained the most trusted mode of evaluation for these tasks. In this work, we show how one can decrease the costs of incorporating human feedback through the design of appropriate statistical estimators.

First, we consider the *"finite incompleteness"* setting where the output space is too large to exhaustively annotate, but we may still expect significant overlap between the output of different systems. Naively combining annotations from different systems leads to a representation bias. Here, we show that the cost of obtaining human feedback can be significantly amortized by using a novel importance-reweighted estimator. We apply this estimator to design a new evaluation methodology for knowledge base population and empirically show that the cost of evaluating precision and recall within this framework can be reduced by a factor of 4.

Next, we consider the *"infinite incompleteness"* setting wherein few, if any, systems ever produce identical output. Traditionally, the community has relied on similarity-based automatic metrics such as BLEU or ROUGE to compare the outputs produced by different systems. Unfortunately, these metrics have been shown to correlate poorly with human judgment and thus introduce bias in evaluation. We derive an unbiased estimator that optimally combines these automatic metrics with human feedback. Our theoretical results allow us to characterize potential cost reductions only in terms of the tasks' subjectivity, measured by inter-annotator variance, and the automatic metrics' quality, measured by correlation with human judgments. On two popular natural language generation tasks, question answering and summarization, we empirically show that currently we can achieve at most a 7–13% reduction in cost on two tasks, exposing fundamental limitations in current automatic metrics.

Finally, we turn our attention to *incompleteness in the training data*, particularly in low-resource settings. Here, our machine learning systems simply have not seen sufficient training data for a particular phenomenon to accurately make predictions for them at test time. To tackle this incarnation of incompleteness, we train a system "on-the-job" by requesting for human feedback in real-time while the model is deployed to economically fill in holes in the training data and thus resolve uncertainty in the model. Our key idea here is to cast the problem as a stochastic game based on Bayesian decision theory, which allows us to balance latency, cost, and accuracy objectives in a principled way. When tested on three classification tasks—named-entity recognition, sentiment classification, and image classification—we obtain an order of magnitude reduction in cost compared to full human annotation even when starting from zero training examples, while also boosting performance relative to a classical supervised model on the expert-provided labels.

Acknowledgments

I would like to thank…

Contents

List of Tables

List of Figures

Chapter 1

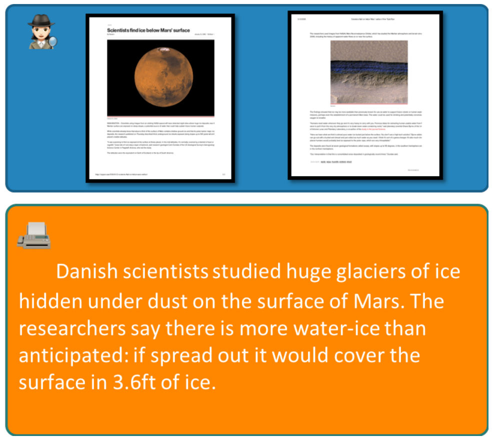
Introduction

One of the most exciting applications of natural language processing (NLP) ahead of us is the development of systems that will help us digest the vast amount of information being generated around the world. For example, we will one day be able to rely on text summarization systems to condense the most salient information from one or more related articles into as few words as necessary (Figure 1.1(a)). We will be be able to get targeted summaries through open-ended question answering systems (Figure 1.1(b)). Finally, knowledge base population (KBP) systems will be able to read large document troves (or the Internet) and present succinct, structured, summaries about the people and organizations mentioned within (Figure 1.1(c)).

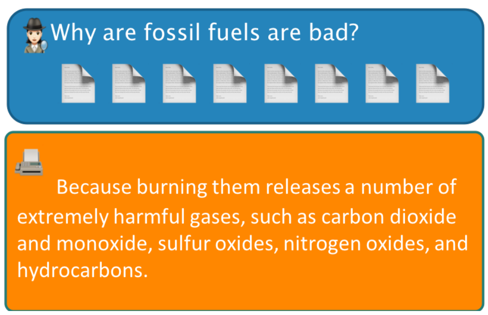
Indeed, this broad vision of what we call *information summarization* has been the goal of many natural language processing systems dating as far back as the 1950s (Luhn, 1958). Research has shown that having access to document summaries significantly improves user satisfaction and their ability to complete fact-gathering tasks (Mani et al., 1999; Mckeown et al., 2005). Building these systems, however, remains a challenge. Today, we are seeing a resurgence in interest in information summarization systems, with over 150 papers published at top NLP conferences in just the last year (2017–18).[1]

Despite years of effort, it is unclear whether we are actually making forward progress

[1]This number was estimated by looking at papers with topics pertaining to information extraction, question answering and text summarization systems.



(a) Text summarization



(b) Open question answering

(c) Knowledge base population

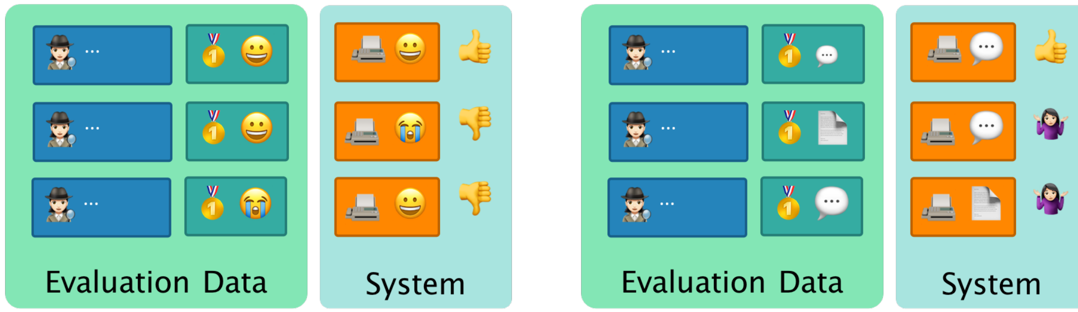
Figure 1.1: An overview of some information summarization tasks, listed in increasing specificity: (a) Text summarization: seeks to identify the most salient information contained within one (or more) article(s) and describe this information in as few words as necessary. (b) Open-response question answering: provides very-targeted summaries that address a specific question. (c) Knowledge base population: reads documents from a large corpus and generates linked entity-centric summaries for every person and organization mentioned within.

on these tasks. Take the task of text summarization for example: Brandow et al. (1995) conducted a large scale human evaluation of different text summarization systems and found that the baseline of simply taking the lead-paragraph significantly outperformed automatic systems. Ten years later, Passonneau et al. (2005) report that half of the systems participating in the DUC-2005 text summarization challenge did worse than the baseline. Even today, we find that recent "state-of-the-art" neural network based text summarization systems fare poorly on human evaluations of language quality relative to simpler extractive systems. The performance of knowledge base population systems as tracked by the TAC KBP challenge has been similarly halting. While there are many reasons for this slow progress, we believe that the lack of an effective evaluation methodology remains one of the most important ones.

The ultimate goal of this thesis is to enable the development of better information summarization systems by addressing systemic problems in how we evaluate them. In this work, we attribute the inherent *incompleteness* of existing evaluation datasets as the key bottleneck to reliably measuring the performance of information summarization systems. We posit that collecting on-demand human feedback is necessary to economically obtaining meaningful measures of system quality by resolving this incompleteness. Our key technical contribution is applying techniques from statistical estimation to reliably integrate human feedback in a cost-effective manner.

1.1 The incompleteness of static evaluation sets

At its core, the prevalent evaluation methodology for most tasks in natural language processing, including the aforementioned information summarization tasks, relies on a evaluation dataset that contains pairs of query inputs and expected outputs (Figure 1.2). On classification problems, e.g. sentiment classification or topic identification, the output typically belongs to a small closed class (e.g. positive, neutral or negative sentiment). When a system also predicts an output belonging to this closed class, we know whether or not it made a mistake by comparing if its prediction is the same as the expected class; Thus, in these settings, we can measure the quality of different systems on the classification task by



(a) Classification (b) Open-response

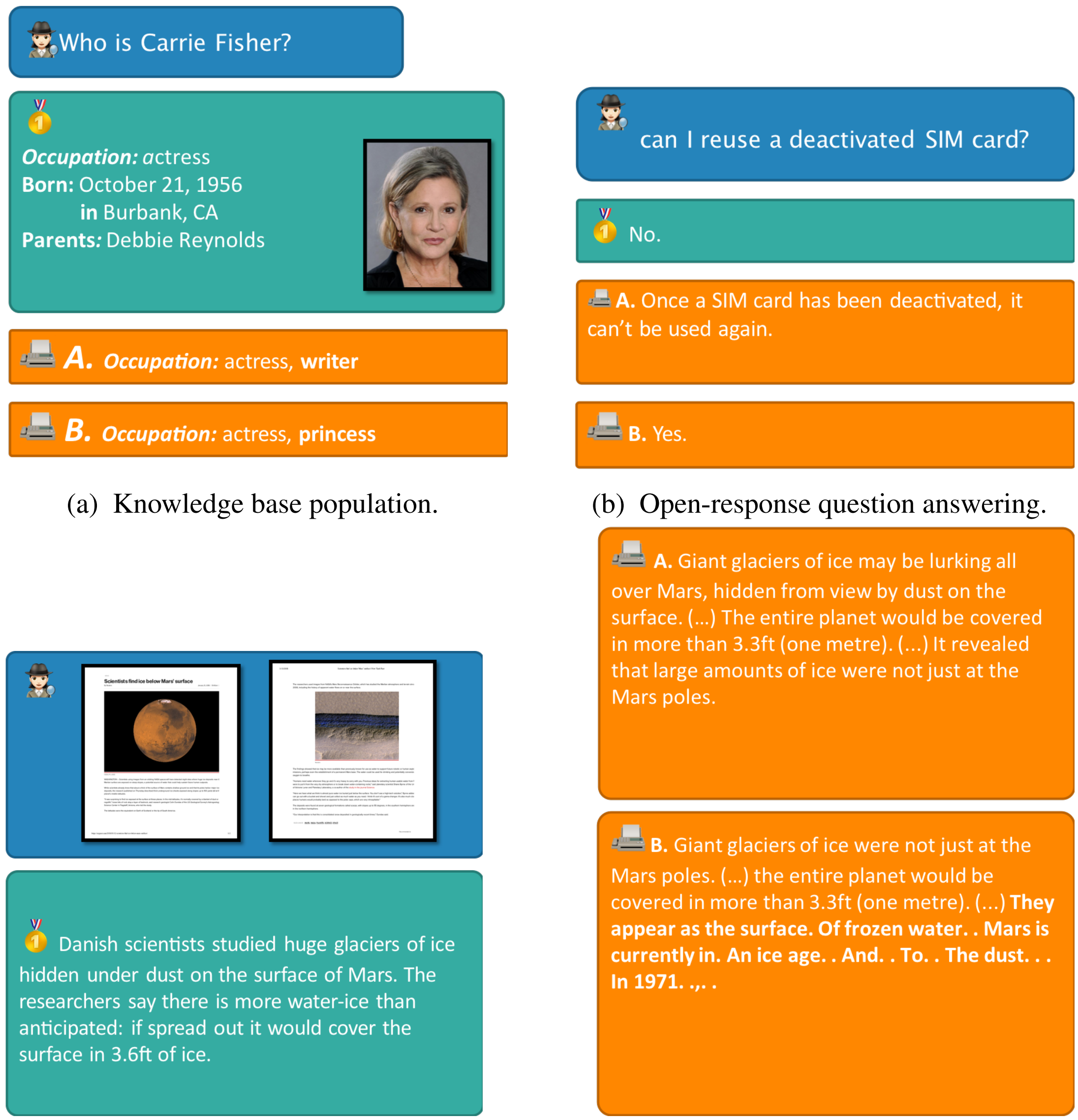
Figure 1.2: Much of our evaluation methodology relies on an evaluation dataset that contains paired inputs (represented by blue boxes) and outputs (green boxes). (a) When the outputs belong to a closed class, e.g. in sentiment classification, it is easy to compare the output of a system (orange boxes) with the reference answer and hence evaluate its performance. (b) On the other hand, in the information summarization tasks we are interested in, there are many possible correct answers. As a result, a system that produces a response that does not match the reference answer is not necessarily incorrect; we simply don't know how to evaluate it. In this sense, the evaluation data is *incomplete*.

simply collecting a sufficiently large evaluation dataset.[2]

Unfortunately, in the information summarization tasks we've seen above, the desired output is not simply a class label but rather an arbitrary piece of text (e.g. in text summarization and open-response question answering) or an arbitrarily large collection of facts (e.g. for KBP). The assumption that there exists a unique knowable correct output simply does not hold. In this sense, the static evaluation dataset we discussed earlier *cannot* contain every possible correct answer and as a result is *incomplete*. Let's look at some examples of how incompleteness manifests in practice and what its consequences on evaluation could be.

Figure 1.3a shows the candidate output of two KBP systems for Carrie Fisher, the late actress who played Princess Leia in the Star Wars movie franchise. The incomplete KBP reference data only specifies that Ms. Fisher is an actress and thus the evaluation is unable to judge whether the system that identifies her to also be an author (which is correct) is better or worse than one that identifies her to also be a princess (which is incorrect): by default, the

[2]Of course, care must be taken to ensure the evaluation dataset collected is reflective of the end-goal for the task.



(a) Knowledge base population.

(b) Open-response question answering.

(c) Text summarization.

Figure 1.3: Examples where the reference data is unable to evaluate a system's response: (a) Knowledge base population: here, the incomplete reference data is unable to identify that it is a true that Carrie Fisher was also an author, but not that she is a 'princess'. (b) Open-response question answering: it is common for systems to output a paraphrase of the reference answer and thus to be judged incorrect despite reporting the right answer. (c) Text summarization: systems rarely produce output that is identical to the reference answer and word-overlap based similarity measures often prefer bad responses with high lexical similarity over good responses that more more lexically distinct.

evaluation penalizes both decisions equally. As a consequence, using this evaluation as an empirical guide leads researchers to avoid improvements that would identify Ms. Fisher as an actress. In Section 3.3 we'll empirically estimate the bias caused by this incompleteness and show that it is often larger than most model improvements.

Next, consider an example in question answering: here, the fundamental problem is that there are many ways to express the same answer (Figure 1.3b). It is not yet easy to automatically identify whether two phrases mean the same thing and hence fairly judge the output systems produce. More subtly, we show that this evaluation is biased towards easier questions with short answers that are more likely to be reproduced by a system: model improvements that improve answers for harder questions will be ignored by the evaluation.

Finally, consider the evaluation of text summarization systems: a system-generated summary will never exactly match the one in the evaluation dataset. A common practice in the community is to use a word-overlap based similarity score such as BLEU (Papineni et al., 2002) or ROUGE (Lin and Rey, 2004). Unfortunately, these automatic metrics have been shown to correlate extremely poorly with human judgment (Novikova et al., 2017). Figure 1.3c shows an example of a published system that learns to game this metric by appending seemingly random words at the end of a summary.

1.2 Addressing incompleteness with human feedback

We've just seen how the incompleteness of our evaluation sets can introduce bias. How fundamental is incompleteness? Broadly, we consider two ends of the spectrum: when the incompleteness is large but "finite" and when the incompleteness is truly "infinite".

In the finite setting, it is possible, though perhaps not economical, to exhaustively annotate the output space. Consider knowledge base population as an example: here it is possible to minimize the impact of incompleteness by exhaustively annotating a sufficiently large document collection of about 100,000 documents, an effort that we estimate would cost at least $1 million. Of course, any adjustment in the task definition (e.g. the relation schema) would require a fresh batch of annotations. In this sense, handling finite incompleteness can be viewed as a prudent cost-saving measure. On the other hand, the number of correct wordings for an answer in text summarization or open-ended question answering are nearly endless; we doubt that any size of static datasets would be sufficient to ameliorate the problem here. We consider the incompleteness in such settings to be infinite.

The crux of the problem is estimating the impact of instances that are unknown to the automatic evaluation that relies on a static dataset. We exploit the fact that the answers to these instances are obvious to humans and propose asking people for feedback *on-demand* using advances in crowdsourcing. In this work, we advocate moving from annotating data in batches to annotating data *on-demand*.

Apart from fixing the problem of incompleteness, incorporating human feedback has several ancillary benefits. First, it ensures that our evaluation does not diverge from the end goals of the task. Second, it provides *qualitative* guidance to help us identify opportunities for improvements.

1.3 Integrating human feedback with statistical estimators

While human evaluation is often regarded as a gold standard for evaluation, it is also considered to be too expensive to be used as when iterating on models. The key question this thesis tries to answer then is: can we reduce the cost of human annotation in evaluation while maintaining its fidelity? Our core contribution is in addressing the problem of incompleteness for two extremes: problems where incompleteness is large but finite (KBP), and problems where incompleteness is truly infinite (text summarization and open-response question answering). In both settings, we hold *unbiased* estimation as our bar: the methods we describe are guaranteed to produce the same results that an exhaustive human evaluation would have.

Amortizing costs when incompleteness is finite. In settings where incompleteness is finite, it is likely that two different systems produce an overlapping set of outputs, suggesting that we may can leverage human annotations obtained for one system to evaluate another. The key challenge is guarding against what we call representation bias: we don't want to skew the evaluation towards a particular class of systems simply because our evaluation data came from these systems. We tackle this problem in Chapter 3 using a novel importance-reweighted estimator. We apply the estimator to evaluate KBP systems and show that we are able to reduce the cost of obtaining human annotations by a factor of 4.

Finding limitations when incompleteness is infinite. On the other end of the spectrum, in tasks like text summarization or open-ended question answering, the output produced consists of free form text: it is extremely unlikely that two systems will ever agree on the output they produce. Here, it seems natural to rely on some "similarity" measure that may allow us to match two similar, but non-identical responses. In Chapter 4, we derive an optimal estimator to combine such a similarity metric with human feedback based on control variates (Owen, 2013). Our theoretical analysis allows us to characterize when it is possible to reduce human annotation costs while guaranteeing unbiasedness. We show that for both text summarization and open-ended question answering current cost savings are modest, about 10–15%, owing to both the poor quality of existing automatic metrics and the inherent annotator variance for these subjective tasks.

Filling in holes in the training data with on-the-job learning. Thus far, we have focused on the problem of incompleteness during evaluation, but it is often the case that *the training data that systems use is also incomplete*, particularly when it is small. We turn to this problem in Chapter 5, where we use the system itself *at test-time* to identify when it is uncertain about an example and then collect human feedback to fill in the holes in its training data. We call this setting "on-the-job learning" because our model learns as inputs arrive: the human feedback is used to train the model and as the model improves over time, the reliance on crowdsourcing queries decreases. We cast the problem as a stochastic game and use Bayesian decision theory to balance latency, cost, and accuracy objectives in a principled way. Unfortunately, computing the optimal policy is intractable, so we develop an approximation based on Monte Carlo Tree Search. We tested our approach on three datasets—named-entity recognition, sentiment classification, and image classification—and show that on-the-job learning can reduce the cost of deploying a high-accuracy system by an order of magnitude when compared to full human annotation even when starting from zero training examples.

1.4 Thesis outline

The rest of the thesis is structured as follows: In Chapter 2, we'll cover the necessary statistical prerequisites to understand what bias means and how different evaluation methodologies can be quantitatively compared. We will also review some background in the different evaluation strategies that have been adopted in natural language processing and the goals that each seeks to meet. In Chapter 3, we will study finite incompleteness using knowledge base population as our motivating example. We'll see how incompleteness can be measured and how we can correct for it by combining on-demand human annotations with a novel statistical estimator. In Chapter 4, we move over to study infinite incompleteness using open question answering and text summarization as motivating examples. We'll see how automatic metrics can be optimally combined with human feedback to reduce the costs of human evaluation. Unfortunately, we'll show that in practice cost savings are modest: However, our optimality result lets us step back and observe that our empirical results actually highlight fundamental limitations in using automatic metrics for unbiased evaluation. In Chapter 5, we step back and look at how incompleteness can be addressed while training a system: we propose a new learning paradigm, "on-the-job learning", that allows the model to query for human feedback when it is not confident in its predictions and thus fill in gaps in its training data. Finally, in Chapter 6, we conclude this thesis with a discussion on the further uses of human feedback in building natural language systems. Our discussion touches upon possible opportunities to improve upon human evaluation as a methodology for natural language generation tasks and how human feedback provides us with a more holistic view of evaluation.

Chapter 2

Background

We begin this chapter by looking at the rich history of evaluation methodologies within the field of NLP in Section 2.1. Next, in Section 2.2, we review key concepts from statistics, like bias and variance, that will allow us to formally design evaluation methodologies.

2.1 Evaluation in NLP: a brief history

Defining an evaluation methodology provides a necessary framework for systematic progress, but can also over-simplify what it means to solve a problem: for example, while perplexity has long served as an evaluation metric for language modeling, it also does not capture the entirety of what it means to truly understand language. Unsurprisingly, this has led to an ongoing debate in the field of artificial intelligence on how to balance philosophical ideals, like genuine language understanding, with pragmatism, like having a single quantitative indicator. In this section, we revisit some of these discussions within the natural language processing field to provide some context for how the future of evaluation should look like.

2.1.1 Early ideas

Even as far back as 1956, before the development of many automated natural language processing systems, the future evaluation of these techniques was discussed. Miller and

Beebe-Center (1956) is perhaps one of the first such papers to explore what modes of evaluation exist for machine translation, beyond asking humans for subjective input. On the automated metric front, the paper introduces lexical overlap based scores that still underpin the most popular evaluation metrics today. The authors recognized that automatic metrics were effective "at the lower end of the scale", but were unsure if it would remain differentiating between better systems. As a result, they also looked at including humans in more objective evaluations, such as computing information gain by asking people to perform the Shannon test after reading the system generated translation, but concluded this might be far too laborious. Finally, the authors also proposed using reading comprehension tests as an objective evaluation of whether the summary was able to correctly convey information: an exciting idea that is seeing recent interest, but with the challenge that picking the right questions can also be quite hard.

2.1.2 An era of shared tasks

As early as the 1960s, the information retrieval community established a shared quantitative evaluation through the Cranfield tests (Cleverdon, 1962, 1967). Despite being carried out through entirely clerical means and without any computerization whatsoever, the evaluation methodology used in the Cranfield tests still continues today (Voorhees, 2007). In it systems were objectively compared according to quantitative metrics using a fixed test collection, consisting of documents, queries and assessments of which documents are relevant for which queries. The Cranfield methodology finally brought empiricism to the field of indexing theory and its findings contradicted the prevailing wisdom: simple single term indices were found to outperform concept-based and thesaurus-based methods. It is not controversial to say that the Cranfield tests, followed by the SMART project (Salton and Lesk, 1965) and later TREC efforts (Harman, 1992) laid the foundations for the information retrieval field as it is today.

Unfortunately, it took another 30 years for the rest of the natural language processing community to adopt a shared evaluation methodology that allowed methods to be objectively compared with each other. The following two quotes from a now-infamous letter by John Pierce (Pierce, 1970), titled "Whither Speech Recognition?", speak to the sordid state of progress in the automated speech recognition community during this period:[1]

> The typical recognizer ... builds or programs *an elaborate system that either does very little or flops in an obscure way*. A lot of money and time are spent. **No simple, clear, sure knowledge is gained**. The work has been an experience, not an experiment.
>
> We are safe in asserting that speech recognition is attractive to money. The attraction is perhaps similar to the attraction of schemes for turning water into gasoline, extracting gold from the sea, curing cancer, or going to the moon. One doesn't attract thoughtlessly given dollars by means of schemes for cutting the cost of soap by 10%. To sell suckers, one uses deceit and offers glamour. [...] **It is clear that glamour and any deceit in the field of speech recognition blind the takers of funds as much as they blind the givers of funds**. Thus, we may pity workers whom we cannot respect.

Following the condemning report, funding for human language technologies dried up for almost 20 years. It was not until 1985 when the DARPA program, lead by Charles Wayne, was able to resume funding for automatic speech recognition by devising the shared task format: systems would be compared using objective metrics on a shared dataset run by a neutral agent, National Institute for Standards and Technology (NIST) and would have to reveal their methods to other participants. The goal of this format was to both guard against the glamour and deceit that had so plagued the field and to ensure that "simple, clear, sure knowledge" was gained. Needless to say, the format worked and has been successfully applied to a number of other tasks in the 1990s through the TIPSTER program. For example, the Text REtrieval Conference (TREC) was set up in 1992 to evaluate information retrieval systems and the Message Understanding Conference (MUC) was set up in the same period to evaluate information extraction systems. Later, the SUMMAC program extended the set of tasks to summarization. It is hard to understate the enormous impact such shared tasks, and the rigor they brought to evaluation, had on the field.

[1] The following quotes were sourced from Mark Liberman's presentation at the CATS reproducibility workshop.

A more detailed discussion of the impact of shared evaluation methodologies is outside the scope of this thesis. Rather, we will focus on how these programs designed their respective evaluation methodologies. While the MUC tasks preferred setting up fully supervised tasks with simple objective evaluation criteria like precision and recall, these methods are harder to apply in information retrieval where it is not feasible to fully label the entire document corpus. As a result, there has been a long running debate about evaluation at TREC. Webber (2010) provides an excellent summary of this work. Perhaps the most interesting line of work on evaluation was in the TIPSTER summarization program (Mani et al., 1999): systems were compared on an end-task based metric of how much they could speed up decision-making for analysts. Later iterations of the summarization task in the Document Understanding Conference (DUC) used more general quality metrics like the quality and coverage of the generated summaries.

At the same time, the machine translation community also developed its own evaluation methodologies through the APRA MT program (White et al., 1994). Initially, systems were compared by asking people to complete a multiple choice reading comprehension task using human generated and system generated translations. Additionally, human panels were used to rate the adequacy and fluency of generated translations.

Unfortunately, the extensive human evaluations made it laborious to compare systems, particularly during development. This motivated the development of automatic metrics, starting with BLEU (Papineni et al., 2002). The automatic metric was proposed as a way to evaluate systems in the aggregate, with the hope that individual errors made by using a simple lexical overlap would be washed out as many different summaries were combined. In the summarization community, ROUGE (Lin and Rey, 2004) was proposed as an efficient way to compute *coverage* of the generated summary. When first reported, it was observed that both the BLEU and ROUGE metrics had had high correlation with human ratings when measured at the system level. Unfortunately, over time, it was found that these metrics were inadequate when systems began to use a more diverse vocabulary (Lavie and Denkowski, 2009; Cohan and Goharian, 2016)

This prompted development of variants that used additional linguistic information or word frequencies, e.g. METEOR (Lavie and Denkowski, 2009; Denkowski and Lavie, 2014) or CiDER (Vedantam et al., 2015). Despite these developments, Liu et al. (2016b) and Novikova et al. (2017) find that a whole suite of automatic evaluation metrics—including those mentioned above—correlate incredibly poorly with human judgment across different datasets and systems, with Pearson's ρ between 0 and 0.3.

Another approach has been to use multiple references, which tends to improve correlation with humans. Indeed, Toutanova and Brockett (2016) report that multiple reference variants of metrics significantly score higher than others, though this can also be low. Unfortunately, there is no clear understanding on how many references are sufficient: Culy and Riehemann (2003) report needing at least 4 reference translations to effectively employ BLEU to a given correlation. Lavie and Denkowski (2009) find that more references helped correlation for METEOR but only marginally. Halteren and Teufel (2003) required up to 40–50 reference factoid summaries for stable consensus.

Finally, the community has also tried to tuning evaluation metrics to optimize for human correlation (Lavie and Denkowski, 2009; Denkowski and Lavie, 2014; Lowe et al., 2017a). While these approaches do increase the correlation for systems in the tune set, their correlations with new systems can be substantially smaller: Lowe et al. (2017a), for example, find that correlations for held-out systems have an average Pearson correlation of 0.13, almost a third of the average correlation of systems that were part of the tune set (0.37).

2.2 Experiment design and statistical analysis

The central theme of this thesis is that we can quantitatively evaluate deeper concepts of understanding by querying humans. Some natural questions that arise are: how many people should we ask, how much can we trust the quantitative measurements we obtain and does it matter whom we ask or when we ask for feedback? In this section, we'll cover the basic statistics necessary to answer these questions.

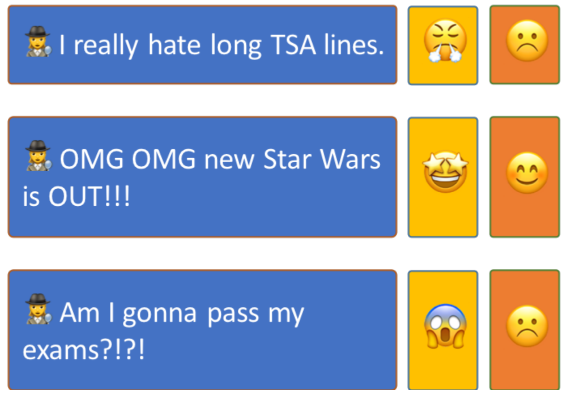


Figure 2.1: When constructing a test collection, for example for sentiment classification, one may trade off collecting fine grained labels (such as "frustration", "excitement" or "fear", illustrated in the figure using emoji) for more general ones (such as "sad" or "happy") that may be easier or more objective to collect.

2.2.1 What makes for a good evaluation procedure?

Instead of diving into the abstract criteria that make an evaluation procedure good, let's begin by sketching out the predominant evaluation procedure: **test collection based evaluation**. Put simply, a **test collection** is a set of inputs and expected outputs on which every system is compared. Each system is assigned a quantitative score based on its performance on the test collection, and different systems are compared on this single quantitative score. In this subsection, we'll look at how we should construct the test collection, and how the quantitative scores should be computed and compared.

Test collections. There are many useful ways to define and use a test collection. Let us take comparing sentiment classification systems as a simple case study (Figure 2.2). Sentiment is inherently multi-faceted (one can express happiness, fear, optimism, etc.) and graded (one can be happy, joyful, ecstatic, etc.), but we may choose to focus simply on distinguishing between positive and negative sentiment because they are more objectively identified by people. On the other hand, if most systems already distinguish between binary sentiment classes, this evaluation will not be useful in comparing or ranking them. An additional benefit of simple binary classification is that the test collection can be used to quantitatively measure the quality of the system in question by comparing the system predictions with expected output (e.g., with accuracy). Unfortunately, this assumption does not hold for the tasks that will be discussed in this thesis: working around this assumption is the key technical contribution of this work. However, for the purposes of intuition, we will assume that the quality or correctness of every output can be perfectly measured using the test collection in this chapter.

Ideally, one would like the test collection to be representative of the inputs that would be found in real life so that the test scores **faithfully** reflect the systems true performance. At the same time, we'd like the test collection to be sufficiently large so that we can be confident in the **reliability** of our conclusions. Of course, the evaluation designer must weigh these considerations with cost or ease of use.

Next, we will cast the concepts of faithfulness, and reliability into statistical terms.

Faithfulness as unbiasedness. Once we have a test collection $T = \{(x, y)\}$, we can run our systems on it. An evaluation procedure compares the gold answers Y with the system's predictions $\hat{Y}$: given a system S, $\hat{y} \in \hat{Y} = S(x)$ for a pair $(x, y) \in T$. When does the evaluation procedure *faithfully* measure what it's supposed to?

In this chapter we will assume that given the triple $z = (x, y, \hat{y})$, where $(x, y) \in T$ are a input-answer pair and $\hat{y} = S(x)$ is a system prediction, we have a quantitative measure of performance, $f(x, y, \hat{y})$ or $f(z)$. One example would be accuracy: $f(x, y, \hat{y}) = 1$ iff $y = \hat{y}$, and 0 otherwise. Let $Z = \{(x, y, S(x)) | (x, y) \in T\}$ be the test collection combined with the systems predictions.

We would like the evaluation procedure to take us from measuring performance on a single example to a more indicative measure of *typical* performance. Ideally, we would like to know how well the system works in the *real world*, i.e., given a some *hypothetical test distribution* of inputs $\mathcal{Q}(x, y)$, we would like to measure performance under the corresponding distribution of system predictions, $\mathcal{P}(x, y, \hat{y}) = \mathcal{Q}(x, y)\mathbf{I}[S(x) = \hat{y}]$. Let the average on this test distribution be $\mu \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{P}}[f(z)] = \frac{1}{|\mathcal{Z}|}\sum_{z \in \mathcal{Z}} \mathcal{P}(z) f(z)$, where $\mathcal{Z}$ is the support of $\mathcal{P}$. We can formalize what it means to be faithful using the concept of unbiasedness: does the evaluation procedure, on average, predict the same measure of performance as μ?

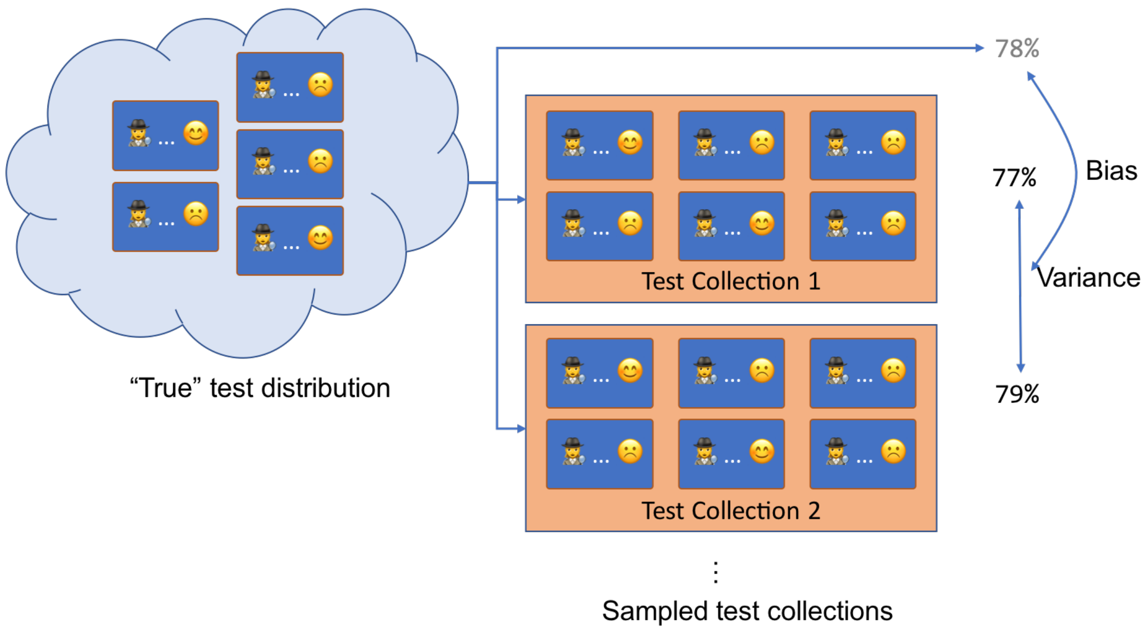


Figure 2.2: Ideally, a test collection is representative of the true test distribution of examples that occur in the true world. When evaluating systems, we would like the performance we measure on our specific test collection to match with what we would get on the true test distribution. While the values we measure on any particular test collection may vary, our goal is to eliminate any discrepancy, or bias, between the average over all test collections and the test distribution. A good evaluation procedure will have less variance on the performance measure across different test collections.

Let $\hat{\mu}(Z)$ be an estimation algorithm that uses this set to predict μ. We say that $\hat{\mu}$ is an **unbiased estimator** of μ if

$$\mathbb{E}_{Z\sim\mathcal{P}}[\hat{\mu}(Z)] = \mu,$$

for any test collection distribution $\mathcal{P}$. One simple method to evaluate μ on the test collection is to simply take the *average* of system predictions on it:

$$\hat{\mu}(Z) = \frac{1}{n}\sum_{z\in Z} f(z).$$

It is easy to see that $\hat{\mu}$ is unbiased *if* T was collected in an unbiased manner, e.g. through random sampling:

$$\begin{aligned}
\mathbb{E}_{Z\sim\mathcal{P}}[\hat{\mu}(Z)] &= \mathbb{E}_{Z\sim\mathcal{P}}\left[\frac{1}{n}\sum_{z\in Z} f(z)\right] \\
&= \mathbb{E}_{z\sim\mathcal{P}}[f(z)] \\
&= \mu.
\end{aligned}$$

We note that unbiasedness requires our estimators to be unbiased not just on the test collection at hand, but for any hypothetical test collection distribution. This is a strong condition that seems natural in the context of evaluation: we would like to be able to trust our procedure irrespective of the type of output our systems produce. At the same time, this condition also presents fundamental limits on the (sample) efficiency of our evaluation procedure, which we will discuss later.

Reliability and variance. Unbiasedness alone is not sufficient for a good evaluation procedure: in principle, using a test collection with just a single point would still be unbiased even though its predictions would vary greatly depending on which point was chosen! Intuitively, the size of our test collection tells us how much *variance* we might expect in our estimate $\hat{\mu}(Z)$ if we had chosen a different set of test examples. We would like our estimate to be indicative of the true performance of our system, as opposed to its performance on just our test collection. This brings us to the second question we must answer: "how big should our test collection be in order to *reliably* use our evaluation procedure"?

Suppose that the variance of $f(z)$ when using a *single point* is σ_f^2, then elementary statistics (Casella and Berger, 1990) tells us that given test collection with n *independently* drawn samples, the variance of $\hat{\mu}$, $\mathrm{Var}[\hat{\mu}] = \frac{\sigma_f^2}{n}$. Furthermore, if we have sufficiently many examples, then the central limit theorem applies and we can say that with high probability that the true performance estimate μ will be fairly close to our observed estimate $\hat{\mu}$ Formally, we have that with probability at least $1 - \delta$ and as $n \to \infty$,

$$\begin{aligned} |\mu - \hat{\mu}| &\leq 2F(\delta)\,\mathrm{Std}[\hat{\mu}] \\ &\leq 2F(\delta)\frac{\sigma_f}{\sqrt{n}}, \end{aligned}$$

where $F(\delta)$ is the Gaussian CDF.[2]

This biggest takeaway from this formula is that the *reliability* of our estimate of the true performance μ is a function not only of the number of samples we have n, but also the intrinsic variation of the system's performance and underlying data-generating distribution, σ_f^2. When picking a test collection, it is helpful to consider the worse case variance for σ_f^2; for example, if $f(z)$ is accuracy and only takes the values of 0 or 1, the worst case variance is $\frac{1}{4}$. As a simple rule of thumb, the number of samples we need to be sure with 95% confidence ($\delta = 0.05$), is about $(\frac{1}{\epsilon^2})$: if we want to the true answer to be be within 1% of the estimate, we fundamentally need about $10,000$ samples.

Comparing estimation procedures The primary way to compare two unbiased estimators is to compare their variances: if an estimator is able to combine samples more efficiently than the mean estimator above, we would be able to get equally reliable or precise estimates of performance with fewer samples. We'll cover when this is possible in the next subsection.

Practical confidence intervals with the bootstrap In practice, we do not actually know what the value of σ_f^2 is: the best we can do is to compute the sample variance on our

[2]This bound can be refined in the finite sample regime, e.g. if we know that $\hat{\mu}$ is distributed as a Gaussian or Binomial random variable. In practice, we will use the empirical bootstrap (described later) to compute confidence bounds without needing to make such assumptions.

test collection, Z. The empirical bootstrap allows us to compute confidence intervals without having to assume any Gaussianity. The procedure is simple and should be computed whenever performances are reported:

1. Suppose we have a test collection Z of size n. Let $\hat{\mu}_0 \stackrel{\text{def}}{=} \hat{\mu}(Z)$ be the estimate on this set.

2. Construct 1,000 to 10,000 new test collections $Z^{(i)}$ by drawing n samples from Z *with replacement*.

3. On each of these collections, evaluate $\hat{\mu}_i \stackrel{\text{def}}{=} \hat{\mu}(Z^{(i)})$ and $\delta_i = \hat{\mu}_i - \hat{\mu}_0$

4. Then the 80% confidence interval for μ is $[\hat{\mu}_0 - \delta_{(0.1)}, \hat{\mu}_0 - \delta_{(0.9)}]$, where $\delta_{(0.1)}$ and $\delta_{(0.9)}$ are respectively the 90th percentile and 10th percentile samples of δ_i.

2.2.2 Additional considerations

Now that we've seen the basic definitions of unbiasedness and variance, we will explore some more nuanced statistical concepts: Are there fundamental limitations on variance of an estimation procedure? Can we measure multiple test distributions with the same set of samples? Are there settings in which biased estimation is appropriate?

Fundamental limitations on the variance of estimators. When comparing two unbiased estimators, what we are really comparing is their variances. Fortunately, there are several important theoretical results that provide guarantees on when a particular estimator is optimal, i.e. has the least variance among all other estimators.

The first of these results is the *Rao-Blackwell theorem*, which states that any estimator $g(X)$ that depends on data X has strictly reduced variance by using sufficient statistics $T(X)$: in other words, $g(T(X))$ will always have equal or less variance than $g(X)$. Unpacking this statement a bit, given a parametrized distribution $\Pr(x|\theta)$ that depends on θ, a statistic t computed from the data is *sufficient* if the conditional probability of the data given t does not depend on θ: $\Pr(x|t, \theta) = \Pr(x|t)$. Some popular examples include the mean of a normal distribution with known variance for which the sample mean is a sufficient statistic. Another way of interpreting sufficiency in this example is that given the sample mean, no more information regarding the normal distribution can be obtained from the sample. Rao-Blackwell tells us that the minimum variance estimator for any function of the mean must depend only on the sample mean. If we would like our unbiased estimator to be optimal (have the least variance possible), we must rely on sufficient statistics.

Conveniently, the *Fisher-Neyman factorization theorem* completely characterizes sufficient statistics. Given a parametric distribution $\Pr(x|\theta)$, T is a sufficient statistic if and only if non-negative functions g and h can be found such that $\Pr(x|\theta) = h(x)g_\theta(T(x))$: the distribution can be factored into a component that depends only on the data and one that depends on the parameters and the sufficient statistic but not the data as a whole. In Chapter 4, we will use the Fisher-Neyman factorization theorem to prove that the unbiased estimators we propose rely on sufficient statistics and use this property to prove strong guarantees about its optimality.

The Rao-Blackwell theorem only states that the optimal estimator must depend on the sufficient statistics of the distribution, but doesn't let us compare different estimators that all only depend on sufficient statistics. Next, we have the *Lehmann-Scheffé theorem*, which states that any estimator that is unbiased for a given unknown quantity and depends on the data *only* through a *complete sufficient statistic* is the unique best estimator of that quantity in that is has least variance among all other distributions. Completeness of a statistic is a much stronger condition that requires distinct values of the statistic to correspond to distinct distributions. Unfortunately, not all distributions or settings have complete statistics. As a result, Lehmann-Scheffé theorem and uniform minimum variance unbiased (UMVU) estimation, though powerful has limited applicability and we do not use it in this thesis. Instead, in Chapter 4, we will use the weaker notion of *minimax optimality*. A minimax optimal estimator has the least maximum variance among all distributions of the data, even though it may have a higher variance on a particular distribution.

Measuring multiple objectives through importance sampling. One of the important criteria for unbiasedness we described in the previous subsection was ensuring that the test collection was sampled in an unbiased manner. In practice, there are situations in which it is hard to collect a completely random sample or situations in which we would like to use same set of samples to measure multiple objectives, for example the accuracy of a sentiment classification system on a particular subset of documents. In these situations, importance sampling can be a useful method to "adjust" the samples from one distribution to another.

Let $q(z)$ be the distribution under which samples were drawn, and let $p(z)$ be the distribution under which we would like to evaluate $\mathbb{E}_p[f(z)]$. Given a set of samples Z drawn from q, an importance sampling estimator *reweights* each sample from q with the weight $\frac{p(z)}{q(z)}$:

$$\mathbb{E}_q[\frac{p(z)}{q(z)}f(z)] = \sum_{z\in\mathcal{Z}} q(z)\frac{p(z)}{q(z)}f(z) \tag{2.1}$$

$$= \sum_{z\in\mathcal{Z}} p(z)f(z) \tag{2.2}$$

$$= \mathbb{E}_p[f(z)]. \tag{2.3}$$

In order to use importance sampling, we must ensure that $q(z) > 0$ whenever $p(z) > 0$. Additionally, the closer that $q(z)$ is to $p(z)$, the less variance we will have in our estimate.

Going beyond unbiasedness. The main objective of estimators in this thesis is to be unbiased. While this is a natural and appealing condition for evaluation, it can be too restrictive of a condition and may require too many samples to be practical. Indeed, it is well known in the statistics literature that there are biased estimators that result in much lower mean square error. One example of this is the James-Stein estimator, a biased estimator of the mean of Gaussian vectors that has uniformly lower mean squared error than the standard unbiased mean estimator. Unfortunately, most biased estimators, including the James-Stein estimator, require some prior knowledge of which test distributions are most likely to significantly reduce variance. When it is reasonable to do so in the evaluation setting, such biased estimators could be considered.

2.3 Finite and infinite incompleteness

The common theme of this thesis is addressing *incompleteness* in training and evaluation datasets with on-demand human feedback. In this section, we will expand on the intuition we presented in Chapter 1 to provide a more formal description of what incompleteness is.

Intuitively, incompleteness arises because the data we are able to observe (the test collection) does not capture phenomena in the data we wish to evaluate (the system's predictions). Formally, let $\mathcal{X}$ be our universe of objects (e.g. relational triples in KBP, answers to questions or summaries of articles), let $X \subset \mathcal{X}$ be the observable data and let $Y \subset \mathcal{X}$ be the target data that we wish to evaluate. Our goal is to measure some aggregate property f (e.g. accuracy, precision, etc.) of Y, $\mu_f = \mathbb{E}_{p(Y)}[f(y)] \stackrel{\text{def}}{=} \sum_{y \in Y} p(y) f(y)$, where $p(y)$ is a given distribution or measure over $y \in Y$. However, because we can only evaluate f on X (the observable set) the value of f on the subset $Y \setminus X$ is indeterminable. As a result, we say that X is **incomplete** when measuring f on Y if Y is not contained in its support, i.e., $Y \nsubseteq X$.[3] While the definition of incomplete really pertains to the test collection X, metric f and evaluation set Y, in the rest of this thesis, we will say that a *task exhibits incompleteness* if its standard test collection is incomplete when measuring the task metric for an arbitrary system.

Finite incompleteness. We say that X is finitely incomplete when measuring f on Y if the cardinality of the universe, $|\mathcal{X}|$, is finite. As an example, consider the setting of KBP: Here, $\mathcal{X}$ consists of relation triples defined by spans in the document corpus, X is a static test collection, Y is a system's predictions that we are trying to evaluate and f is precision or recall. While there can be a very large number of such triples, the number is finite because the document collection is finite. With problems of finite incompleteness, it is fundamentally possible to annotate all of $\mathcal{X}$ and thus ensure that $Y \subseteq X = \mathcal{X}$.

Infinite incompleteness. Likewise, we say that X is infinitely incomplete when measuring f on Y if the cardinality of the universe, $|\mathcal{X}|$, is infinite. As an example, consider the setting of text summarization where $\mathcal{X}$ consists of all possible text strings which is infinite,

[3] An equivalent, more rigorous, definition of incompleteness can be given in terms of measure theory: we say that X is *incomplete* when measuring f on $p(Y)$ if $p(Y \setminus X) > 0$.

X is the set of reference summaries, Y is the set of system-generated summaries and f is the quality of the summary.

Addressing incompleteness with on-demand annotation. The main solution for incompleteness we present in this thesis is on-demand human annotation. Following the formal description above, on-demand annotation "cheats" by annotating elements of $\mathcal{X}$ as required, effectively making all of $\mathcal{X}$ observable and thus stepping around the problem of incompleteness. The main issue then is to reduce the costs of measuring μ_f, and this will be the objective of the technical solutions we present through the rest of this thesis.

Chapter 3

Importance-reweighted estimators for unbiased on-demand evaluation of knowledge base population

Knowledge base population (KBP) systems take in a large document corpus and extract entities and their relations. Thus far, KBP evaluation has relied on judgements on the pooled predictions of existing systems because it is prohibitively expensive to exhaustively annotate the entire document corpus. Cast in the framework of finite and infinite completeness we introduced in Chapter 1, KBP is a perfect exemplar of finite incompleteness. In this chapter, we show that this evaluation is problematic: when a new system predicts a previously unseen relation, it is penalized even if it is correct. This leads to significant bias against new systems, which counterproductively discourages innovation in the field.

We then introduce our approach to correcting the bias: Our first contribution is a new importance-sampling based evaluation which corrects for this bias by annotating a new system's predictions on-demand via crowdsourcing. We show this eliminates bias and reduces variance using data from the 2015 TAC KBP task. Our second contribution is an implementation of our method made publicly available as an online KBP evaluation service. We pilot the service by testing diverse state-of-the-art systems on the TAC KBP 2016 corpus and obtain accurate scores in a cost effective manner.

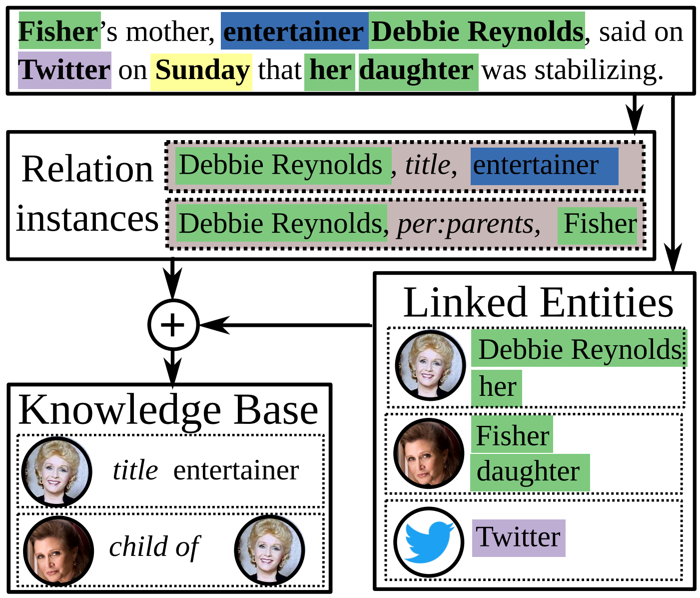


Figure 3.1: An example describing entities and relations in knowledge base population.

3.1 Introduction

Harnessing the wealth of information present in unstructured text online has been a long standing goal for the natural language processing community. In particular, knowledge base population seeks to automatically construct a knowledge base consisting of relations between entities from a document corpus. Knowledge bases have found many applications including question answering (Berant et al., 2013; Fader et al., 2014; Reddy et al., 2014), automated reasoning (Kalyanpur et al., 2012) and dialogue (Han et al., 2015).

Evaluating these systems remains a challenge as it is not economically feasible to exhaustively annotate every possible candidate relation from a sufficiently large corpus. As a result, a pooling-based methodology is used in practice to construct datasets, similar to them methodology used in information retrieval (Jones and Rijsbergen, 1975; Harman, 1993). For instance, at the annual NIST TAC KBP evaluation, all relations predicted by participating systems are pooled together, annotated and released as a dataset for researchers to develop and evaluate their systems on. However, during development, if a new system predicts a previously unseen relation it is considered to be wrong even if it is correct. The discrepancy between a system's true score and the score on the pooled dataset is called *pooling bias* and is typically assumed to be insignificant in practice (Zobel, 1998).

The key finding of this chapter contradicts this assumption and shows that the pooling bias is actually significant, and it penalizes newly developed systems by 2% F_1 on average (Section 3.3). Novel improvements, which typically increase scores by less than 1% F_1 on existing datasets, are therefore likely to be clouded by pooling bias during development. Worse, the bias is larger for a system which predicts qualitatively different relations systematically missing from the pool. Of course, systems participating in the TAC KBP evaluation do not suffer from pooling bias, but this requires researchers to wait a year to get credible feedback on new ideas.

This bias is particularly counterproductive for machine learning methods as they are trained assuming the pool is the complete set of positives. Predicting unseen relations and learning novel patterns is penalized. The net effect is that researchers are discouraged from developing innovative approaches, in particular from applying machine learning, thereby slowing progress on the task.

Our second contribution, described in Section 3.4, addresses this bias through a new evaluation methodology, *on-demand evaluation*, which avoids pooling bias by querying crowdworkers, while minimizing cost by leveraging previous systems' predictions when possible. We then compute the new system's score based on the predictions of past systems using importance weighting. As more systems are evaluated, the marginal cost of evaluating a new system decreases. We show how the on-demand evaluation methodology can be applied to knowledge base population in Section 3.5. Through a simulated experiment on evaluation data released through the TAC KBP 2015 Slot Validation track, we show that we are able to obtain unbiased estimates of a new systems score's while significantly reducing variance.

Finally, our third contribution is an implementation of our framework as a publicly available evaluation service at https://kbpo.stanford.edu, where researchers can have their own KBP systems evaluated. The data collected through the evaluation process could even be valuable for relation extraction, entity linking and coreference, and will also be made publicly available through the website. We evaluate three systems on the 2016 TAC KBP corpus for about $150 each (a fraction of the cost of official evaluation). We believe the public availability of this service will speed the pace of progress in developing KBP

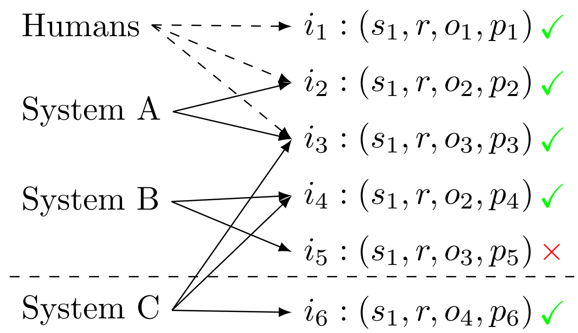


Figure 3.2: In pooled evaluation, an evaluation dataset is constructed by labeling relation instances collected from the pooled systems (A and B) and from a team of human annotators (Humans). However, when a new system (C) is evaluated on this dataset, some of its predictions (i_6) are missing and cannot be fairly evaluated. Here, the precision and recall for C should be $\frac{3}{3}$ and $\frac{3}{4}$ respectively, but its evaluation scores are estimated to be $\frac{2}{3}$ and $\frac{2}{3}$. The discrepancy between these two scores is called *pooling bias*.

systems.

3.2 Background

In knowledge base population, each relation is a triple (SUBJECT, PREDICATE, OBJECT) where SUBJECT and OBJECT are some globally unique entity identifiers (e.g. Wikipedia page titles) and PREDICATE belongm to a specified schema.[1] A KBP system returns an output in the form of *relation instances* (SUBJECT, PREDICATE, OBJECT, PROVENANCE), where PROVENANCE is a description of where exactly in the document corpus the relation was found. In the example shown in Figure 3.1, CARRIE FISHER and DEBBIE REYNOLDS are identified as the subject and object, respectively, of the predicate CHILD OF, and the whole sentence is provided as provenance. The provenance also identifies that CARRIE FISHER is referenced by **Fisher** within the sentence. Note that the same relation can be expressed in multiple sentences across the document corpus; each of these is a different relation instance.

[1] The TAC KBP guidelines specify a total of 65 predicates (including inverses) such as per:title or org:founded_on, etc. Subject entities can be people, organizations, geopolitical entities, while object entities also include dates, numbers and arbitrary string-values like job titles.

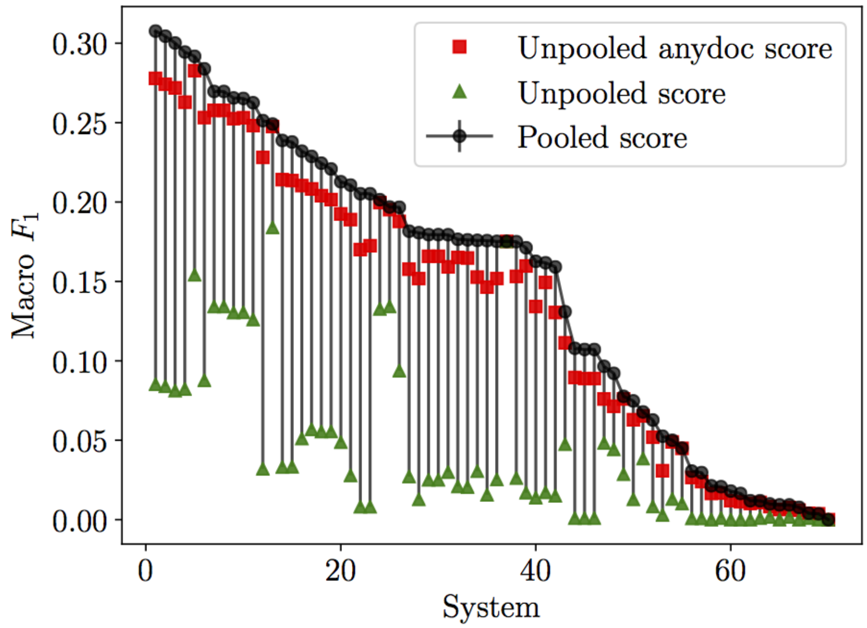
Pooled evaluation. The primary source of evaluation data for KBP comes from the annual TAC KBP competition organized by NIST (Ji et al., 2011). Let E be a held-out set of *evaluation entities*. There are two steps performed in parallel: First, each participating system is run on the document corpus to produce a set of relation instances; those whose subjects are in E are labeled as either positive or negative by annotators. Second, a team of annotators identify and label correct relation instances for the evaluation entities E by manually searching the document corpus within a time budget (Ellis et al., 2012). These labeled relation instances from the two steps are combined and released as the evaluation dataset. In the example in Figure 3.2, systems A and B were used in constructing the pooling dataset, and there are 3 distinct relations in the dataset, between s_1 and o_1, o_2, o_3.

A system is evaluated on the precision of its predicted relation instances for the evaluation entities E and on the recall of the corresponding predicted *relations* (not instances) for the same entities (see Figure 3.2 for a worked example). When using the evaluation data during system development, it is common practice to use the more lenient anydoc score that ignores the provenance when checking if a relation instance is true. Under this metric, predicting the relation (CARRIE FISHER, CHILD OF, DEBBIE REYNOLDS) from an ambiguous provenance like "**Carrie Fisher** and **Debbie Reynolds** arrived together at the awards show" would be considered correct even though it would be marked wrong under the official metric.

3.3 Measuring pooling bias

The example in Figure 3.2 makes it apparent that pooling-based evaluation can introduce a systematic bias against unpooled systems. However, it has been assumed that the bias is insignificant in practice given the large number of systems pooled in the TAC KBP evaluation. We will now show that the assumption is not valid using data from the TAC KBP 2015 evaluation.[2]

[2]Our results are not qualitatively different on data from previous years of the shared task.



	Median bias		
	Precision	Recall	Macro F_1
Official	17.93%	17.00%	15.51%
anydoc	2.34%	1.93%	2.05%

Figure 3.3: Median pooling bias (difference between pooled and unpooled scores) on the top 40 systems of TAC KBP 2015 evaluation using the official and anydoc scores. The bias is much smaller for the lenient anydoc metric, but even so, it is larger than the largest difference between adjacent systems (1.5% F_1) and typical system improvements (around 1% F_1).

Measuring bias. In total, there are 70 system submissions from 18 teams for 317 evaluation entities (E) and the evaluation set consists of 11,008 labeled relation instances.[3] The original evaluation dataset gives us a good measure of the true scores for the participating systems. Similar to Zobel (1998), which studied pooling bias in information retrieval, we simulate the condition of a team not being part of the pooling process by removing any

[3]The evaluation set is actually constructed from compositional queries like, "what does Carrie Fisher's parents do?": these queries select relation instances that answer the question "who are Carrie Fisher's parents?", and then use those answers (e.g. "Debbie Reynolds") to select relation instances that answer "what does Debbie Reynolds do?". We only consider instances selected in the first part of this process.

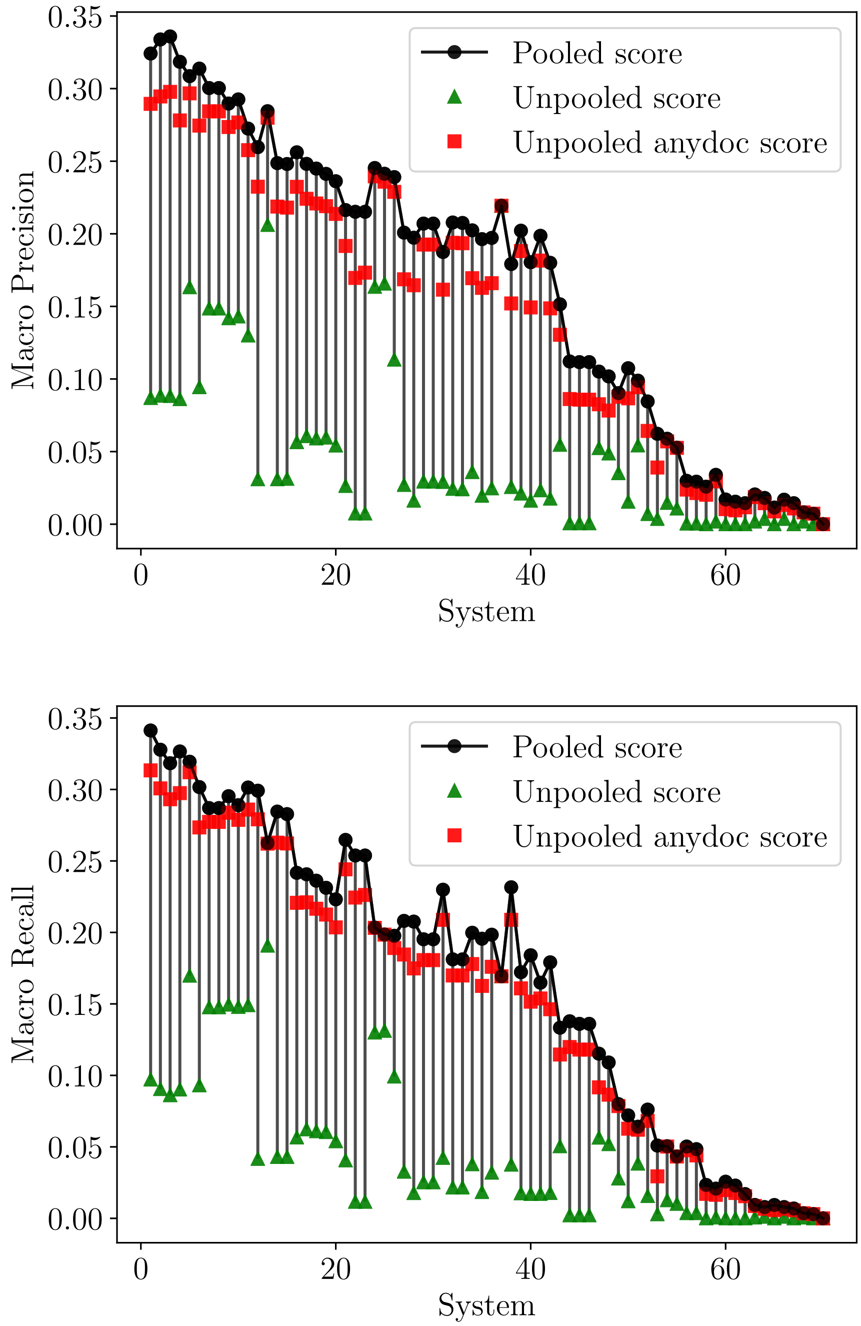


Figure 3.4: Pooling bias broken down by precision and recall.

predictions that are unique to its systems from the evaluation dataset. The pooling bias is then the difference between the true and unpooled scores.

Results. Figure 3.3 shows the results of measuring pooling bias on the TAC KBP 2015 evaluation on the F_1 metric using the official and anydoc scores.[4][5] Figure 3.4 further breaks down the results for precision and recall. We observe that even with lenient anydoc heuristic, the median bias (2.05% F_1) is much larger than largest difference between adjacently ranked systems (1.5% F_1). This experiment shows that pooling evaluation is significantly and systematically biased against systems that make novel predictions!

3.4 On-demand evaluation with importance sampling

Pooling bias is fundamentally a sampling bias problem where relation instances from new systems are underrepresented in the evaluation dataset. We could of course sidestep the problem by exhaustively annotating the entire document corpus, by annotating all mentions of entities and checking relations between all pairs of mentions. However, that would be a laborious and prohibitively expensive task: using the interfaces we've developed (Section 3.6), it costs about \$15 to annotate a single document by non-expert crowdworkers, resulting in an estimated cost of at least \$1,350,000 for a reasonably large corpus of 90,000 documents (Dang, 2016). The annotation effort would cost significantly more with expert annotators. In contrast, *labeling* relation instances from system predictions can be an order of magnitude cheaper than finding them in documents: using our interfaces, it costs only about \$0.18 to verify each relation instance compared to \$1.60 per instance extracted through exhaustive annotations.

We propose a new paradigm called on-demand evaluation which takes a lazy approach to dataset construction by annotating predictions from systems *only when they are underrepresented*, thus correcting for pooling bias as it arises. In this section, we'll formalize the problem solved by on-demand evaluation independent of KBP and describe a cost-effective solution that allows us to accurately estimate evaluation scores without bias using

[4] We note that anydoc scores are on average $0.88\% F_1$ larger than the official scores.

[5] The outlier at rank 36 corresponds to a University of Texas, Austin system that only filtered predictions from other systems and hence has no unique predictions itself.

importance sampling. We'll then instantiate the framework for KBP in Section 3.5.

3.4.1 Problem statement

Let $\mathcal{X}$ be the universe of (relation) instances, $\mathcal{Y} \subseteq \mathcal{X}$ be the unknown subset of correct instances, $X_1, \dots X_m \subseteq \mathcal{X}$ be the predictions for m systems, and let $Y_i = X_i \cap \mathcal{Y}$. Let $X = \bigcup_{i=1}^m X_i$ and $Y = \bigcup_{i=1}^m Y_i$. Let $f(x) \stackrel{\text{def}}{=} \mathbb{I}[x \in \mathcal{Y}]$ and $g_i(x) = \mathbb{I}[x \in X_i]$, then the precision, π_i, and recall, r_i, of the set of predictions X_i is

$$\pi_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_i}[f(x)] \qquad\qquad r_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_0}[g_i(x)],$$

where p_i is a distribution over X_i and p_0 is a distribution over $\mathcal{Y}$. We assume that p_i is known, e.g. the uniform distribution over X_i and that we know p_0 up to normalization constant and can sample from it.

In on-demand evaluation, we can query $f(x)$ (e.g. labeling an instance) or draw a sample from p_0; typically, querying $f(x)$ is significantly cheaper than sampling from p_0. We obtain prediction sets $X_1, \dots, X_m$ sequentially as the systems are submitted for evaluation. Our goal is to estimate π_i and r_i for each system $i = 1, \dots, m$.

3.4.2 Simple estimators

We can estimate each π_i and r_i independently with simple Monte Carlo integration. Let $\hat{X}_1, \dots, \hat{X}_m$ be multi-sets of $n_1, \dots, n_j$ i.i.d. samples from $X_1, \dots, X_m$ respectively, and let $\hat{Y}_0$ be a multi-set of n_0 samples drawn from $\mathcal{Y}$. Then, the simple estimators for precision and recall are:

$$\hat{\pi}_i^{(\text{simple})} = \frac{1}{n_i} \sum_{x \in \hat{X}_i} f(x) \qquad\qquad \hat{r}_i^{(\text{simple})} = \frac{1}{n_0} \sum_{x \in \hat{Y}_0} g_i(x).$$

3.4.3 Joint estimators

The simple estimators are unbiased but have wastefully large variance because evaluating a new system does not leverage labels acquired for previous systems.

On-demand evaluation with the joint estimator works as follows: First $\hat{Y}_0$ is randomly sampled from $\mathcal{Y}$ once when the evaluation framework is launched. For every new set of predictions X_m submitted for evaluation, the minimum number of samples n_m required to accurately evaluate X_m is calculated based on the current evaluation data, $\hat{Y}_0$ and $\hat{X}_1, \ldots, \hat{X}_{m-1}$. Then, the set $\hat{X}_m$ is added to the evaluation data by evaluating $f(x)$ on n_m samples drawn from X_m. Finally, estimates π_i and r_i are updated for each system $i = 1, \ldots, m$ using the joint estimators that will be defined next. In the rest of this section, we will answer the following three questions:

1. How can we use all the samples $\hat{X}_1, \ldots \hat{X}_m$ when estimating the precision π_i of system i?
2. How can we use all the samples $\hat{X}_1, \ldots, \hat{X}_m$ with $\hat{Y}_0$ when estimating recall r_i?
3. Finally, to form $\hat{X}_m$, how many samples should we draw from X_m given existing samples and $\hat{X}_1, \ldots, \hat{X}_{m-1}$ and $\hat{Y}_0$?

Estimating precision jointly. Intuitively, if two systems have very similar predictions X_i and X_j, we should be able to use samples from one to estimate precision on the other. However, it might also be the case that X_i and X_j only overlap on a small region, in which case the samples from X_j do not accurately represent instances in X_i and could lead to a biased estimate. We address this problem by using importance sampling (Owen, 2013), a standard statistical technique for estimating properties of one distribution using samples from another distribution.

In importance sampling, if $\hat{X}_i$ is sampled from q_i, then $\frac{1}{n_i} \sum_{x \in \hat{X}_i} \frac{p_i(x)}{q_i(x)} f(x)$ is an unbiased estimate of π_i. We would like the proposal distribution q_i to both leverage samples from all m systems and be tailored towards system i. To this end, we first define a distribution over systems j, represented by probabilities w_{ij}. Then, define q_i as sampling a j and drawing $x \sim p_j$; formally $q_i(x) = \sum_{j=1}^{m} w_{ij} p_j(x)$.

We note that $q_i(x)$ not only significantly differs between systems, but also changes as new systems are added to the evaluation pool. Unfortunately, the standard importance sampling procedure requires us to draw and use samples from each distribution $q_i(x)$ independently and thus cannot effectively reuse samples drawn from different distributions. To this end, we introduce a practical refinement to the importance sampling procedure: we independently draw n_j samples according to $p_j(x)$ from each of the m systems independently and then numerically integrate over these samples using the weights w_{ij} to “mix” them appropriately to produce and unbiased estimate of π_i while reducing variance. Formally, we define the *joint precision estimator*:

$$\hat{\pi}_i^{\text{(joint)}} \stackrel{\text{def}}{=} \sum_{j=1}^{m} \frac{w_{ij}}{n_j} \sum_{x \in \hat{X}_j} \frac{p_i(x) f(x)}{q_i(x)},$$

where each $\hat{X}_j$ consists of n_j i.i.d. samples drawn from p_j.

It is a hard problem to determine what the optimal mixing weights w_{ij} should be. However, we can formally verify that if X_i and X_j are disjoint, then $w_{ij} = 0$ minimizes the variance of π_i, and if $X_i = X_j$, then $w_{ij} \propto n_j$ is optimal. This motivates the following heuristic choice which interpolates between these two extremes: $w_{ij} \propto n_j \sum_{x \in \mathcal{X}} p_j(x) p_i(x)$.

Estimating recall jointly. The recall of system i can be expressed can be expressed as a product $r_i = \theta \nu_i$, where θ is the *recall of the pool*, which measures the fraction of all positive instances predicted by the pool (any system), and ν_i is the *pooled recall of system* i, which measures the fraction of the pool's positive instances predicted by system i. Letting $g(x) \stackrel{\text{def}}{=} \mathbb{I}[x \in X]$, we can define these as:

$$\nu_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_0}[g_i(x) \mid x \in X] \qquad\qquad \theta \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_0}[g(x)].$$

We can estimate θ analogous to the simple recall estimator $\hat{r}_i$, except we use the pool g instead a system g_i. For ν_i, the key is to leverage the work from estimating precision. We already evaluated $f(x)$ on $\hat{X}_i$, so we can compute $\hat{Y}_i \stackrel{\text{def}}{=} \hat{X}_i \cap \mathcal{Y}$ and form the subset $\hat{Y} = \bigcup_{i=1}^{m} \hat{Y}_i$. $\hat{Y}$ is an approximation of $\mathcal{Y}$ whose bias we can correct through importance reweighting. We then define estimators as follows:

$$\hat{\nu}_i \stackrel{\text{def}}{=} \frac{\sum_{j=1}^{m} \frac{w_{ij}}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x) g_i(x)}{q_i(x)}}{\sum_{j=1}^{m} \frac{w_{ij}}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x)}{q_i(x)}}$$

$$\hat{r}_i^{\text{(joint)}} \stackrel{\text{def}}{=} \hat{\theta} \hat{\nu}_i \quad \hat{\theta} \stackrel{\text{def}}{=} \frac{1}{n_0} \sum_{x \in \hat{Y}_0} g(x).$$

where q_i and w_{ij} are the same as before.

Adaptively choosing the number of samples. Finally, a desired property for on-demand evaluation is to label new instances only when the current evaluation data is insufficient, e.g. when a new set of predictions X_m contains many instances not covered by other systems. We can measure how well the current evaluation set covers the predictions X_m by using a conservative estimate of the variance of $\hat{\pi}_m^{\text{(joint)}}$.[6] In particular, the variance of $\hat{\pi}_m^{\text{(joint)}}$ is a monotonically decreasing function in n_m, the number of samples drawn from X_m. We can easily solve for the minimum number of samples required to estimate $\hat{\pi}_m^{\text{(joint)}}$ within a confidence interval ϵ by using the bisection method (Burden and Faires, 1985).

3.5 On-demand evaluation for KBP

Applying the on-demand evaluation framework to a task requires us to answer three questions:

1. What is the desired distribution over system predictions p_i?
2. How do we label an instance x, i.e. check if $x \in \mathcal{Y}$?
3. How do we sample from the unknown set of true instances $x \sim p_0$?

In this section, we present practical implementations for knowledge base population.

[6] Further details can be found in Appendix A.1 of the supplementary material.

3.5.1 Sampling from system predictions

Both the official TAC-KBP evaluation and the on-demand evaluation we propose use micro-averaged precision and recall as metrics. However, in the official evaluation, these metrics are computed over a fixed set of evaluation entities chosen by LDC annotators, resulting in two problems: (a) defining evaluation entities requires human intervention and (b) typically a large source of variability in evaluation scores comes from not having enough evaluation entities (see e.g. (Webber, 2010)). In our methodology, we replace manually chosen evaluation entities by sampling entities from each system's output according p_i. In effect, p_i makes explicit the decision process of the annotator who chooses evaluation entities.

Identifying a reasonable distribution p_i is an important implementation decision that depends on what one wishes to evaluate. Our goal for the on-demand evaluation service we have implemented is to ensure that KBP systems are fairly evaluated on diverse subjects and predicates, while at the same time, ensuring that entities with multiple relations are represented to measure completeness of knowledge base entries. The dual objectives of balancing the head and tail of the distribution apply broadly other tasks as well, such as information retrieval.

To pick the specific distributions to use in our work, we looked to the evaluation guidelines of the TAC KBP competition developed by the Linguistic Development Consortium (LDC) (Ellis et al., 2015; Mayfield and Finin, 2012) as a well-respected community standard and tried to adapt it to our framework.

The LDC query distribution and metrics. The first aspect we looked into was the distribution of entities that were queried in the official evaluations. The KBP evaluation is well known for using query entities that participate in several relations but are still not common enough to have a knowledge base entry on say Wikipedia. We wanted to capture this property in our evaluation as well.

To get a better sense of the distribution of entities that were queried, we used the entities (i.e., people and organizations) that were recognized by the Stanford KBP entity linking system (Chaganty et al., 2017b) and plotted a log-log distribution of the number of entities that appear in some number of documents (Figure 3.5). As one might expect, the distribution approximately follows a power-law and is extremely long-tailed: there are many

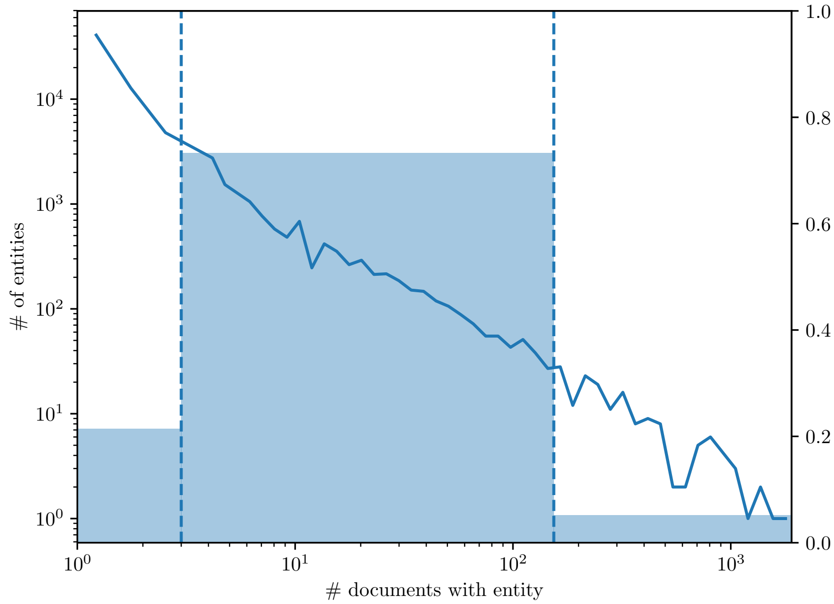


Figure 3.5: The solid line plots a histogram of how many documents a particular entity appeared in the TAC KBP 2016 corpus. The distribution is approximately a power-law distribution. Overlayed is a histogram of the frequency of the actual query entities used in the TAC KBP 2015 evaluation, binned by their frequency. We consider entities that appear in 3 or less unique documents (the 50th percentile) to be “low” frequency entities, those that appear more than 154 unique document (90th percentile) to be “high” frequency entities and those that appear in between to be “medium” frequency entities. The TAC KBP evaluation prefers medium frequency and low frequency entities.

entities that appear in just one or two documents.

We then binned entities into three categories, "low" frequency entities that appear in at most 3 documents in the corpus (this is the 50th percentile), "medium" frequency entities that appear in between 3 and 154 (90th percentile) documents and finally, "high" frequency entities that appear in more than 154 documents. The proportion of query entities in these three bins has been overlayed on top of the distribution in Figure 3.5. We see that a majority of the entities queried are medium frequency, followed by low frequency entities and a handful of high frequency entities. We would like to ensure that our sampling distributions adequately capture these medium frequency entities.

Identifying diverse clustered relations. Finally, we wanted to ensure that the entities we sampled from systems were well distributed both over medium frequency entities and over relations. Unsurprisingly, the naive approach of uniformly sampling relations outputted by systems, is dominated by high frequency entities (Figure 3.6a) on a few common relations (e.g., $\mathrm{per{:}title}$; see Figure 3.7).

Our first attempt at rectifying the problem was to use two distributions that sampled relations inversely proportional to the frequency of their subject entity and relation label respectively:

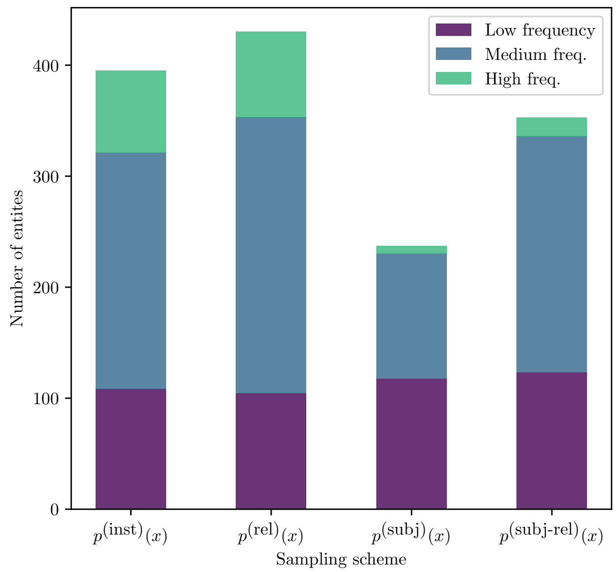
$$p^{(\text{subj})}(x) \propto \frac{1}{|\mathrm{subj}(x)|} \qquad\qquad p^{(\text{rel})}(x) \propto \frac{1}{|\mathrm{rel}(x)|}$$

These distributions solved the two problems individually, but could not be combined because they differed in their support. There was an additional problem: we hardly ever selected two relations from the same entity, as shown in Figure 3.6b. We wanted to maintain this property to be able to test the entity linking component of systems.

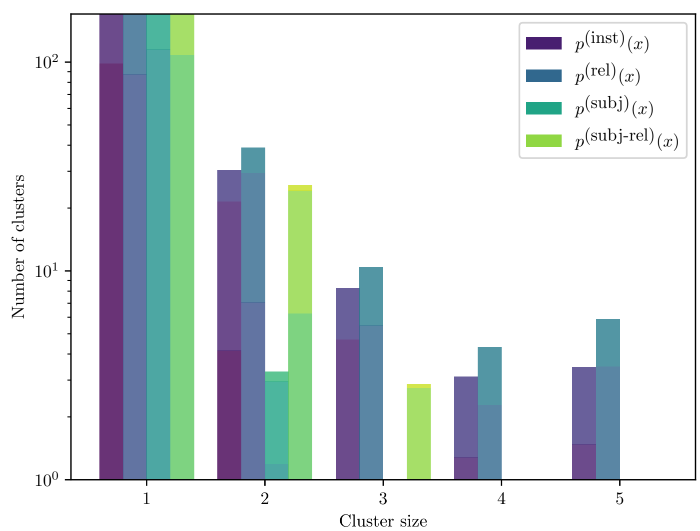
Our final proposed distribution rectified this by combining $p^{(\text{subj})}(x)$ and $p^{(\text{rel})}(x)$ and including a factor for the number of *other* relations the subject entity had.

$$p^{(\text{subj-rel})}(x) \propto \frac{|\mathrm{subj\text{-}relns}(x)|}{|\mathrm{subj}(x)||\mathrm{rel}(x)|},$$

Figure 3.6a shows how this new distribution has both a better representation of medium



(a) Distribution over entity classes



(b) Distribution over entity clusters

Figure 3.6: We compared the types of entities that different relation instance sampling schemes preferred when we sampled 1,000 instances using them. Plots are averaged over 100 draws. (a) In terms of low, medium and high frequency entities, both $p^{(\text{subj})}$ and $p^{(\text{subj-reln})}$ sample fewer high frequency entities which are otherwise over represented. (b) In terms of how many entities were sampled with more than one relation (a cluster) were sampled, $p^{(\text{subj-reln})}$ is still able to capture a comparable number of these unlike $p^{(\text{reln})}$.

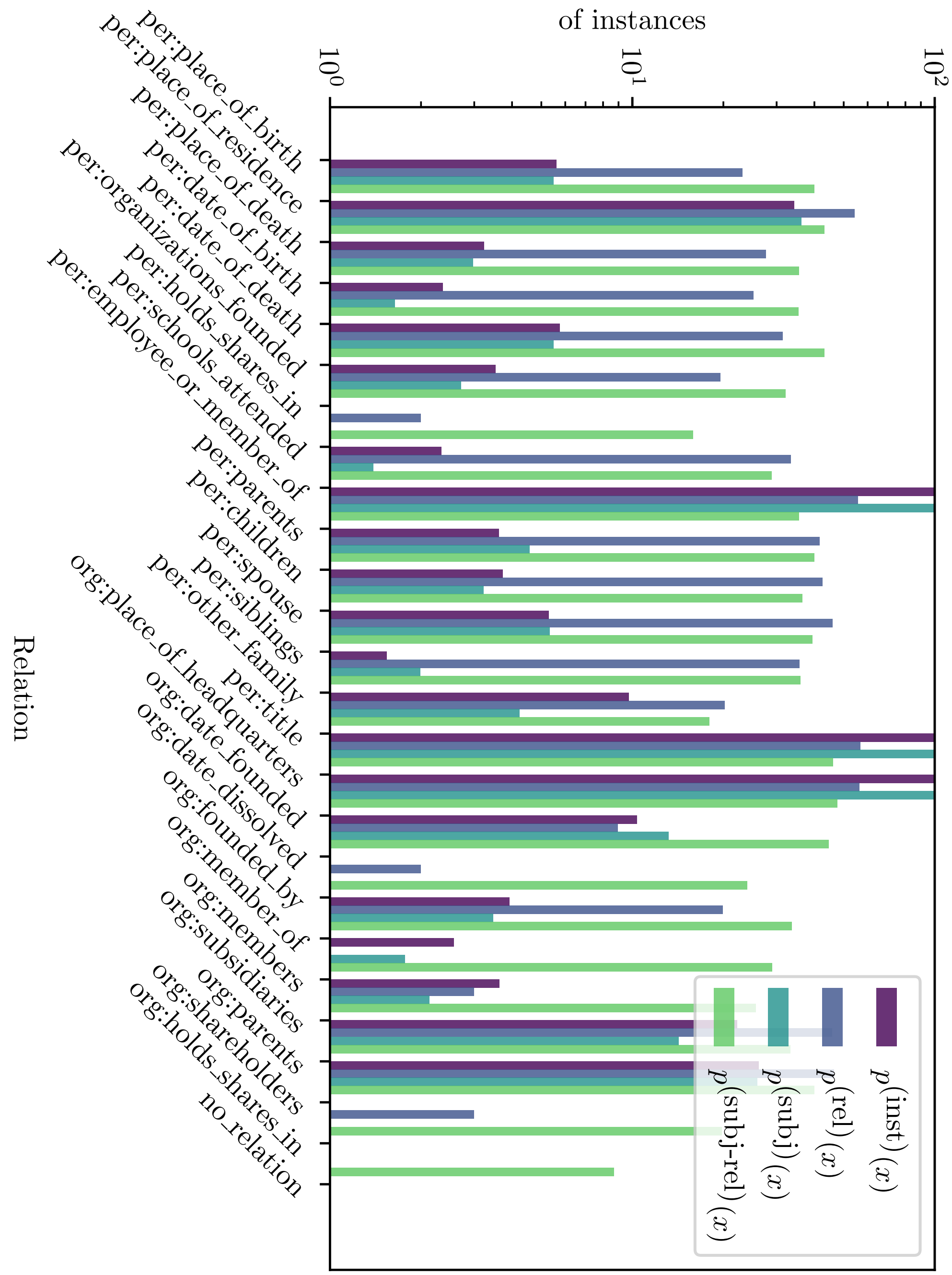


Figure 3.7: We compared the types of relations that different relation instance sampling schemes preferred when we sampled 1,000 instances using them. Plots are averaged over 100 draws. Both $p^{(\text{inst})}$ and $p^{(\text{subj})}$ do not draw any instances for particular relations, e.g. org:shareholders, while $p^{(\text{rel})}$ and $p^{(\text{subj-rel})}$ adequately represent each relation.

frequency entities and is well balanced across all the relations. Figure 3.6b also shows that it is able to represent clusters of relations with the same entity.

3.5.2 Labeling predicted instances

We label predicted relation instances by presenting the instance's provenance to crowdworkers and asking them to identify if a relation holds between the identified subject and object mentions (Figure 3.8a). Crowdworkers are also asked to link the subject and object mentions to their canonical mentions within the document and to pages on Wikipedia, if possible, for entity linking. On average, we find that crowdworkers are able to perform this task in about 20 seconds, corresponding to about $0.05 per instance. We requested 5 crowdworkers to annotate a small set of 200 relation instances from the 2015 TAC-KBP corpus and measured a substantial inter-annotator agreement with a Fleiss' kappa of 0.61 with 3 crowdworkers and 0.62 with 5. Consequently, we take a majority vote over 3 workers in subsequent experiments.

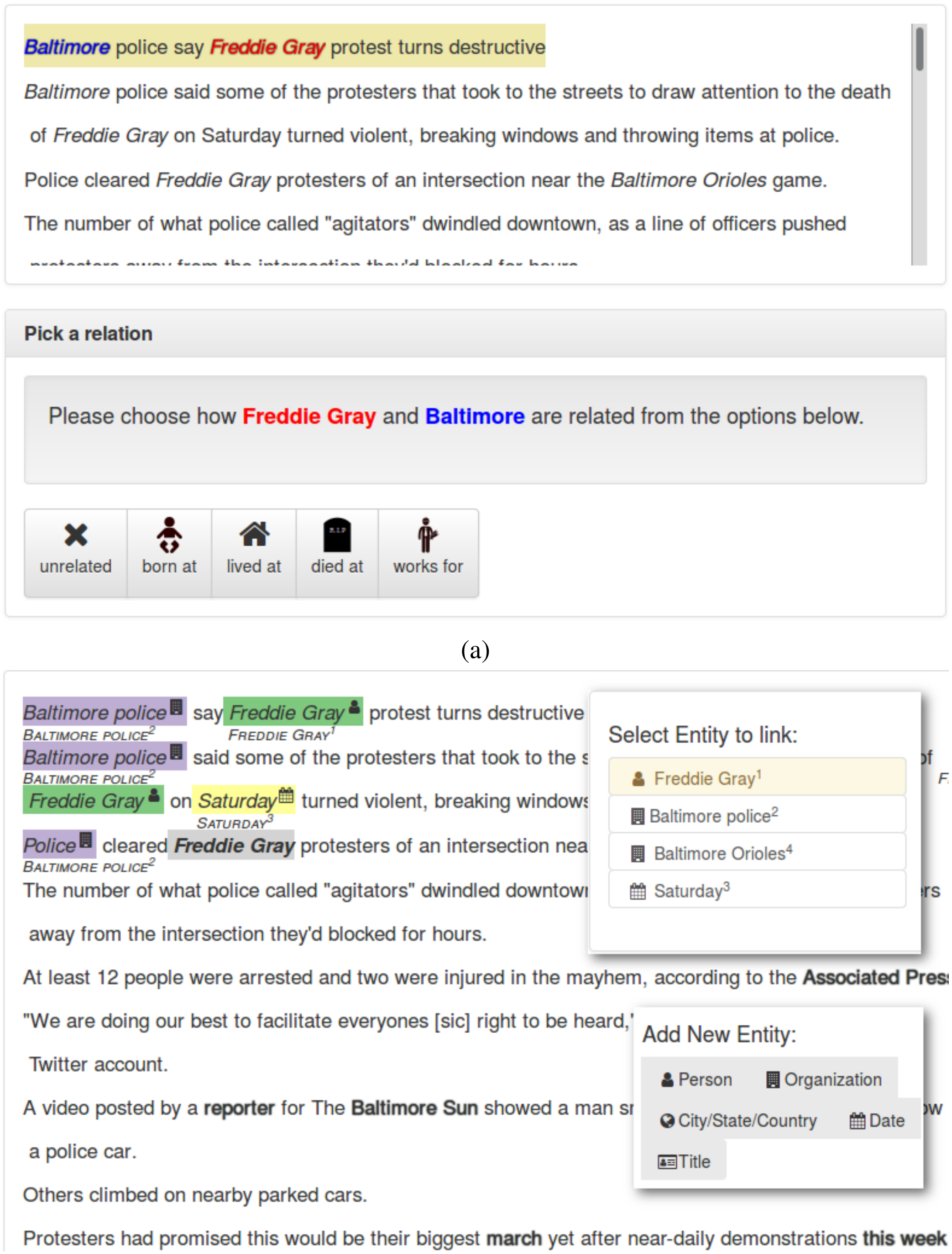
3.5.3 Sampling true instances

Sampling from the set of true instances $\mathcal{Y}$ is difficult because we can't even enumerate the elements of $\mathcal{Y}$. As a proxy, we assume that relations are identically distributed across documents and have crowdworkers annotate a random subset of documents for relations.

Here too, we wanted to come up with a good distribution with which to query documents to exhaustively annotate. It is not enough for us to ensure the documents contain a good distribution of medium frequency entities: we must also ensure that these entities appear in multiple documents to fairly test systems' entity linking components.

First, we tried using documents that were sampled uniformly from the corpus. As Figure 3.9 shows, a randomly sampled document contains a good range of low, medium and high frequency entities, but almost none of these entities are shared across the documents of the collection. We note that the fact that low frequency entities dominate the entity distribution is to be expected because they constitute the majority of entities in any given document,[7] but we can correct for this when sampling relations.

[7] We also tried sampling documents based on the frequency of entities they contained (estimated using the



(a)

(b)

Figure 3.8: Interfaces for annotating (a) relations and (b) entities.

Stanford entity linker), but found that it did not significantly reduce the number of low frequency entities.

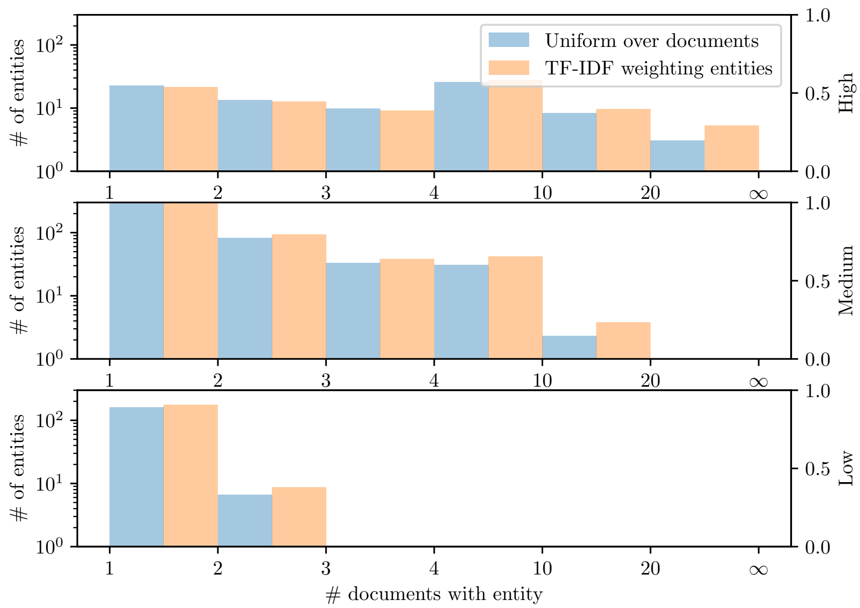


Figure 3.9: In order to properly test the entity linker when measuring recall, an important feature of KBP systems, we must identify documents for exhaustive annotation that contain some entity clusters. A sample of 200 documents uniformly picked from the document collection only exhibits clusters for high frequency entities. Our TF-IDF sampling scheme increases the frequency with which medium and low frequency entities appear across multiple documents in the sample.

Instead, we used a two-step sampling procedure. First, we randomly sample documents for 20% of our exhaustive document collection and then sample the remaining documents proportional to their aggregate TF-IDF scores with the entities identified in the first sample. Figure 3.9 shows the distribution of entities that were sampled by this method, using the Stanford entity linking system to identify entities in the first sample. The method is able to increase the number of medium frequency entities appearing in multiple documents. When implementing this method in practice, we explicitly avoided using the Stanford entity linking system when identifying entities in documents because we were concerned about the bias it would introduce into our recall estimates. Instead, we identified the entities in the first sample using crowdsourcing with our exhaustive annotation interface.

3.5.4 Labeling true instances

Once a random subset of documents was selected, we asked crowdworkers to identify relations using an interface we developed (Figure 3.8b). Crowdworkers begin by identifying every mention span in a document. For each mention, they are asked to identify its type, canonical mention within the document and associated Wikipedia page if possible. They are then presented with a separate interface to label predicates between pairs of mentions within a sentence that were identified earlier.

We compare crowdsourced annotations against those of expert annotators using data from the TAC KBP 2015 EDL task on 10 randomly chosen documents. We find that 3 crowdworkers together identify 92% of the entity spans identified by expert annotators, while 7 crowdworkers together identify 96%. When using a token-level majority vote to identify entities, 3 crowdworkers identify about 78% of the entity spans; this number does not change significantly with additional crowdworkers. We also measure substantial token-level inter-annotator agreement using Fleiss' kappa for identifying typed mention spans ($\kappa = 0.83$), canonical mentions ($\kappa = 0.75$) and entity links ($\kappa = 0.75$) with just three workers. Based on this analysis, we use token-level majority over 3 workers in subsequent experiments.

The entity annotation interface is far more involved and takes on average about 13 minutes per document, corresponding to about \$2.60 per document, while the relation annotation interface takes on average about \$2.25 per document. Because documents vary significantly in length and complexity, we set rewards for each document based on the number of tokens (.75c per token) and mention pairs (5c per pair) respectively. With 3 workers per document, we paid about \$15 per document on average. Each document contained an average 9.2 relations, resulting in a cost of about \$1.61 per relation instance. We note that this is about ten times as much as labeling a relation instance.

3.5.5 Computing scores

The official TAC KBP evaluation computes macro precision, recall and F_1 using the average taken over all entities. In the evaluation, systems are asked to present exactly one justification or provenance for every relation that is ultimately labeled by expert annotators. A relation is only judged correct if the entities in the relation are both correctly linked and entailed by the provided justification. This specification presents a few problems and opportunities for our evaluation protocol that we discuss in this subsection.

Utilizing multiple justifications for a relation. We would like systems to report multiple justifications for a particular relation to ensure that we can maximally reuse our annotations. In principle, we only care if any of the reported justifications are correct. However, we chose to measure the average correctness of all the justifications provided for two reasons: (a) first, we think that looking at average correctness gives a better indicator of system quality and (b) it is hard to adapt a sampling based approach, which implicitly computes an average, to compute the "maximum" accuracy across all the justifications without having to annotate all the justifications.

Handling entity linking. When annotating a relation predicted by a system, we ask annotators to (a) ensure that the subject is correctly linked to a canonical entity within the document and (b) to attempt to link the entity to a Wikipedia page (as a globally canonical entity). However, KBP systems often use their own entity linking scheme that does not correspond to the ones in Wikipedia. As a result, we only handle entity linking *within* a document, ensuring that the canonical entity span reported by the system corresponds to the one indicated by the annotators. We hope to measuring cross-document entity linking in future work.

Factored scores. As part of our evaluation, we independently identify the correctness of the relation between the subject and object in the justification, as well as the correctness of the entity links of the subject and object. As a result, we are able to compute correctness scores for relation classification, entity linking and the combined task. Additionally, we are able to use per-relation and per-entity sampling distributions to report unbiased scores for each relation and entity category. As a result, we are able to automate a basic error analysis method and make it more statistically sound.

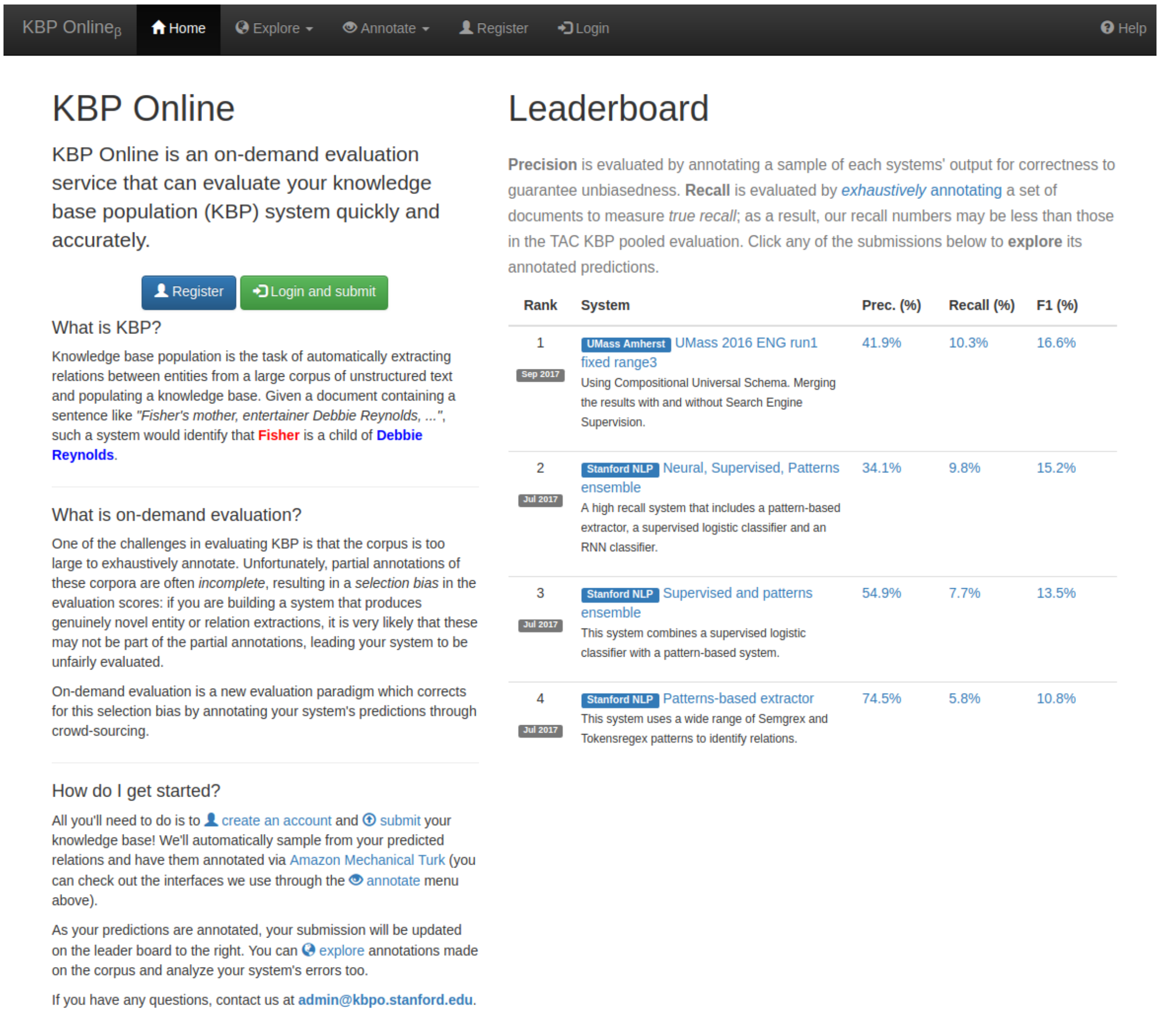
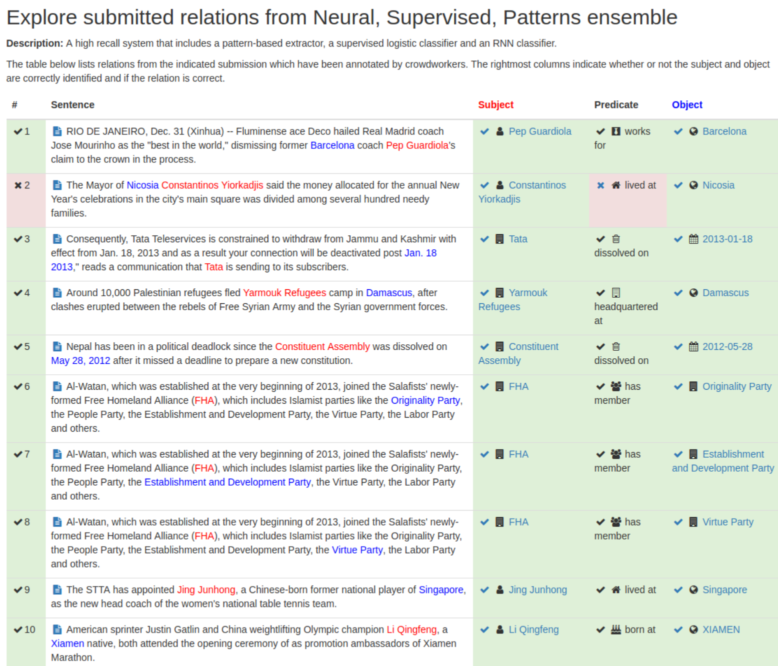
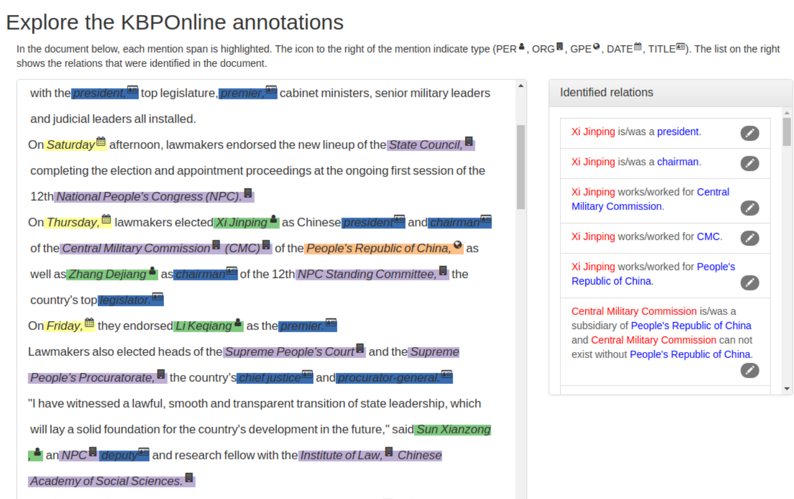


Figure 3.10: A screenshot of the home page of our evaluation service, KBP Online. Our service takes care of the entire pipeline of sampling relation instances, annotating them and computing a score. Once the system is completely evaluated, we show their entry on the home page leaderboard along with an 80% confidence intervals estimated using bootstrap (not shown here).



(a) Annotations of a particular system's predictions



(b) Exhaustive annotations on entire documents

Figure 3.11: Apart from automating the evaluation of KBP systems, KBP Online also allows users to see (a) annotations of their predictions to identify the errors their systems have made and (b) annotations that were made on entire documents to identify relations that no system has been able to extract yet.

3.5.6 Other implementation details

We implemented the online service using the Django web framework with Postgres as our relational database. For performance reasons, we computed sampling distributions and evaluation scores entirely within the database. We used the service to host the crowd-sourcing interfaces described in Section 3.5 and dynamically generated Human Intelligence Tasks (HITs) on Amazon Mechanical Turk from our server. Maintaining exactly which relations were sampled and with what probability is important for our importance-reweighted estimator, and we stored these details in our database. A large amount of effort went into automating the background tasks that collected annotations from Amazon Mechanical Turk, aggregated them and updating the scores.

In terms of an interface to users, they simply need to upload a single file in the official TAC KBP format containing their predicted knowledge base. Our service entirely takes care of computing the required statistics and sampling from the systems. At present, we collect about 500 samples for every system, which corresponds to an 80% confidence interval of $\pm 3\%$. Once the system is completely evaluated, we show the entry on the home page leaderboard (Figure 3.10) along with an 80% confidence intervals estimated using bootstrap (not shown here).

Additionally, we allow users to drill down into annotations we collected on documents and on a system's predictions (Figure 3.11). These annotations help users identify the errors systems are making as well as identify relations that no system is currently able to extract. We hope to include a feature that also provides a quantitative summary of the models performance on different relations and break down errors between relation classification and entity linking. We hope that these features will help push the state of the art of knowledge base population forward.

3.6 Experiments

Let us now see how well on-demand evaluation works in practice. We begin by empirically studying the bias and variance of the joint estimator proposed in Section 3.4 and find it is able to correct for pooling bias while significantly reducing variance in comparison with the simple estimator. We then demonstrate that on-demand evaluation can serve as a practical replacement for the TAC KBP evaluations by piloting a new evaluation service we have developed to evaluate three distinct systems on TAC KBP 2016 document corpus.

3.6.1 Bias and variance of the on-demand evaluation.

Once again, we use the labeled system predictions from the TAC KBP 2015 evaluation and treat them as an exhaustively annotated dataset. To evaluate the pooling methodology we construct an evaluation dataset using instances found by human annotators and labeled instances pooled from 9 randomly chosen teams (i.e. half the total number of participating teams), and use this dataset to evaluate the remaining 9 teams. On average, the pooled evaluation dataset contains between 5,000 and 6,000 labeled instances and evaluates 34 different systems (since each team may have submitted multiple systems). Next, we evaluated sets of 9 randomly chosen teams with our proposed simple and joint estimators using a total of 5,000 samples: about 150 of these samples are drawn from $\mathcal{Y}$, i.e. the full TAC KBP 2015 evaluation data, and 150 samples from each of the systems being evaluated.

We repeat the above simulated experiment 500 times and compare the estimated precision and recall with their true values (Figure 3.12). The simulations once again highlights that the pooled methodology is biased, while the simple and joint estimators are not. Furthermore, the joint estimators significantly reduce variance relative to the simple estimators: the median 90% confidence intervals reduce from 0.14 to 0.06 precision and from 0.14 to 0.08 for recall.

3.6.2 Number of samples required by on-demand evaluation

Separately, we evaluate the efficacy of the adaptive sample selection method described in Section 3.4.3 through another simulated experiment. In each trial of this experiment, we evaluate the top 40 systems in random order. As each subsequent system is evaluated, the number of samples to pick from the system is chosen to meet a target variance and added to the current pool of labeled instances. To make the experiment more interpretable, we choose the target variance to correspond with the estimated variance of having 500 samples. Figure 3.13 plots the results of the experiment. The number of samples required to estimate systems quickly drops off from the benchmark of 500 samples as the pool of labeled instances covers more systems. This experiment shows that on-demand evaluation using joint estimation can scale up to an order of magnitude more submissions than a simple estimator for the same cost.

3.6.3 A mock evaluation for TAC KBP 2016

We have implemented the on-demand evaluation framework described here as an evaluation service to which researchers can submit their own system predictions. As a pilot of the service, we evaluated three relation extraction systems that also participated in the official 2016 TAC KBP competition. Each system uses Stanford CoreNLP (Manning et al., 2014) to identify entities, the Illinois Wikifier (Ratinov et al., 2011) to perform entity linking and a combination of a rule-based system (P), a logistic classifier (L), and a neural network classifier (N) for relation extraction. We used 15,000 Newswire documents from the 2016 TAC KBP evaluation as our document corpus. In total, 100 documents were exhaustively annotated for about $2,000 and 500 instances from each system were labeled for about $150 each. Evaluating all three system only took about 2 hours.

Figure 3.13b reports scores obtained through on-demand evaluation of these systems as well as their corresponding official TAC evaluation scores. While the relative ordering of systems between the two evaluations is the same, we note that precision and recall as measured through on-demand evaluation are respectively higher and lower than the official scores. This is to be expected because on-demand evaluation measures precision using each systems output as opposed to an externally defined set of evaluation entities. Likewise, recall is measured using exhaustive annotations of relations within the corpus instead of annotations from pooled output in the official evaluation.

3.7 Related work

The subject of pooling bias has been extensively studied in the information retrieval (IR) community starting with Zobel (1998), which examined the effects of pooling bias on the

TREC AdHoc task, but concluded that pooling bias was not a significant problem. However, when the topic was later revisited, Buckley et al. (2007) identified that the reason for the small bias was because the submissions to the task were too similar; upon repeating the experiment using a novel system as part of the TREC Robust track, they identified a 23% point drop in average precision scores![8]

Many solutions to the pooling bias problem have been proposed in the context of information retrieval, e.g. adaptively constructing the pool to collect relevant data more cost-effectively (Zobel, 1998; Cormack et al., 1998; Aslam et al., 2006), or modifying the scoring metrics to be less sensitive to unassessed data (Buckley and Voorhees, 2004; Sakai and Kando, 2008; Aslam et al., 2006). Many of these ideas exploit the ranking of documents in IR which does not apply to KBP. While both Aslam et al. (2006) and Yilmaz et al. (2008) estimate evaluation metrics by using importance sampling estimators, the techniques they propose require knowing the set of all submissions beforehand. In contrast, our on-demand methodology can produce unbiased evaluation scores for new development systems as well.

There have been several approaches taken to crowdsource data pertinent to knowledge base population (Vannella et al., 2014; Angeli et al., 2014; He et al., 2015; Liu et al., 2016a). The most extensive annotation effort is probably Pavlick et al. (2016), which crowdsources a knowledge base for gun-violence related events. In contrast to previous work, our focus is on *evaluating systems*, not collecting a dataset. Furthermore, our main contribution is not a large dataset, but an evaluation service that allows anyone to use crowdsourcing predictions made by their system.

3.8 Discussion

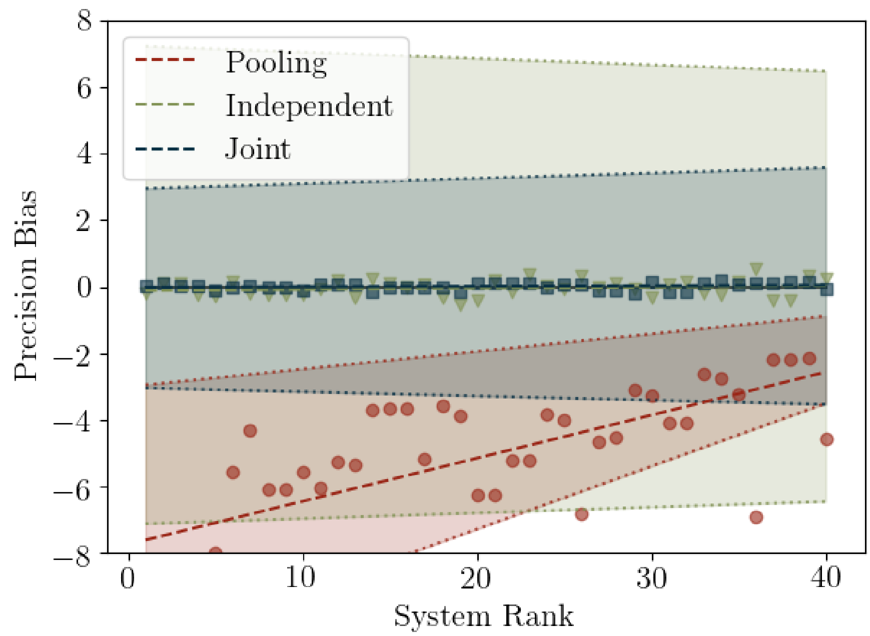
Over the last ten years of the TAC KBP task, the gap between human and system performance has barely narrowed despite the community's best efforts: top automated systems score less than 36% F_1 while human annotators score more than 60%. In this chapter, we've shown that the current evaluation methodology may be a contributing factor because of its bias against novel system improvements. The new on-demand framework proposed in this chapter addresses this problem by obtaining human assessments of new system output

[8]For the interested reader, Webber (2010) presents an excellent survey of the literature on pooling bias.

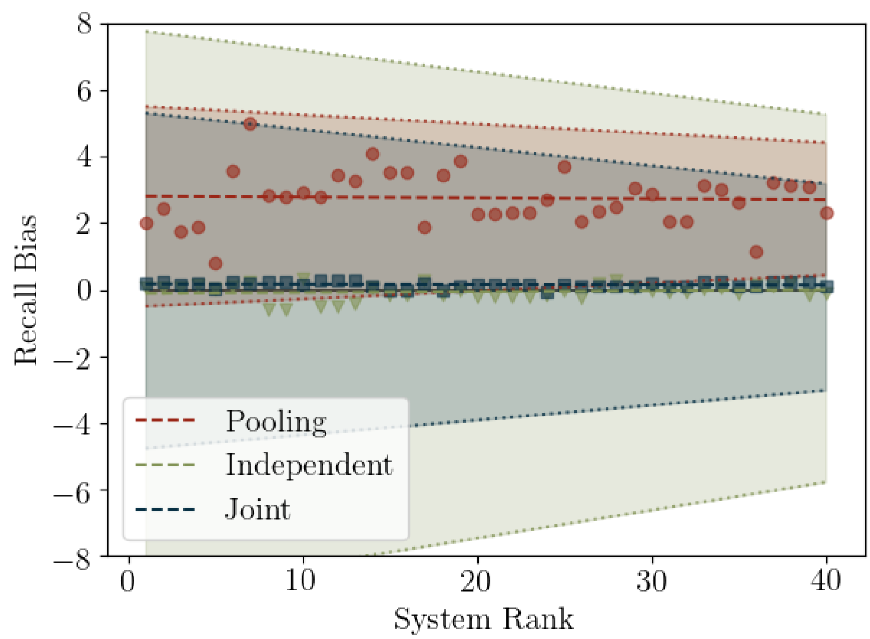
through crowdsourcing. The framework is made economically feasible by carefully sampling output to be assessed and correcting for sample bias through importance sampling.

Of course, simply providing better evaluation scores is only part of the solution and it is clear that better datasets are also necessary. However, the very same difficulties in scale that make evaluating KBP difficult also make it hard to collect a high quality dataset for the task. As a result, existing datasets (Angeli et al., 2014; Adel et al., 2016) have relied on the output of existing systems, making it likely that they exhibit the same biases against novel systems that we've discussed in this chapter. We believe that providing a fair and standardized evaluation platform as a service allows researchers to exploit such datasets and while still being able to accurately measure their performance on the knowledge base population task.

While our observations regarding bias were specifically instantiated and empirically measured for knowledge base population, they apply to any scenario exhibiting finite incompleteness, e.g. information retrieval, event extraction or extractive question answering (different from the open-response question answering we discussed in Chapter 1). Our solution applies wherever precision or recall are being measured across multiple systems and can be easily extended to estimate any other metric that is averaged over instances in the data. However, the extent to which we are able to reduce the variance of estimation, and thereby cost, depends on the overlap between the predictions of two systems: in practice, we expect the overlap to be fairly high in scenarios exhibiting finite incompleteness. Put together, in this chapter we have shown that we can indeed reduce the costs of unbiased evaluation in finite incompleteness settings with importance-reweighted estimators.

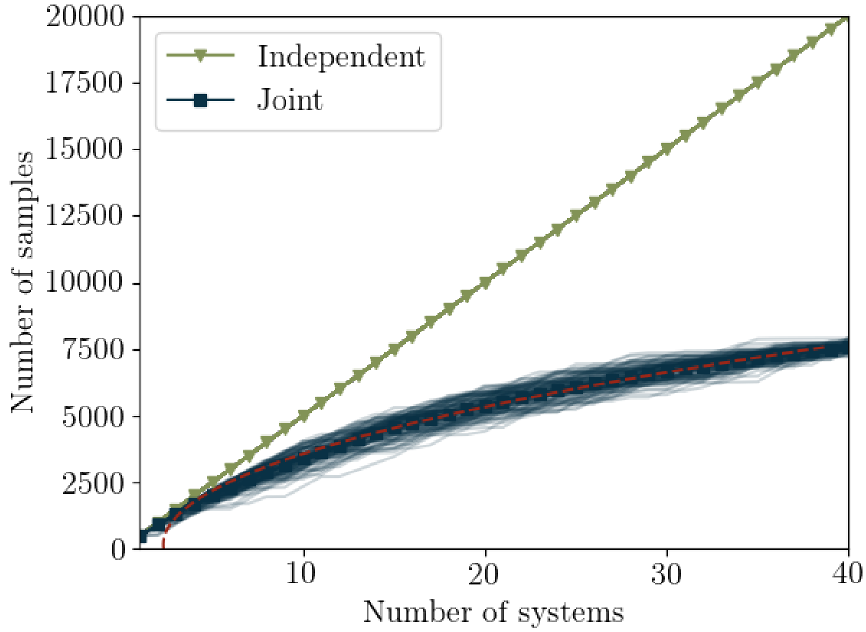


(a)



(b)

Figure 3.12: A comparison of bias for the pooling, simple and joint estimators on the TAC KBP 2015 challenge. Each point in the figure is a mean of 500 repeated trials; dotted lines show the 90% quartile. Both the simple and joint estimators are unbiased, and the joint estimator is able to significantly reduce variance.



(a)

Sys.	P	R	F_1
	TAC KBP evaluation		
P	47.6%	11.0%	17.9%
P+L	35.5%	18.4%	24.2%
P+L+N	26.3%	27.0%	26.6%
	On-demand evaluation		
P	74.7%	5.8%	10.8%
P+L	54.7%	7.6%	13.3%
P+L+N	34.0%	9.8%	15.2%

(b)

Figure 3.13: **(a):** A comparison of the number of samples used to estimate scores under the fixed and adaptive sample selection scheme. Each faint line shows the number of samples used during a single trial, while solid lines show the mean over 100 trials. The dashed line shows a square-root relationship between the number of systems evaluated and the number of samples required. Thus joint estimation combined with adaptive sample selection can reduce the number of labeled annotations required by an order of magnitude. **(b):** Precision (P), recall (R) and F_1 scores from a pilot run of our evaluation service for ensembles of a rule-based system (R), a logistic classifier (L) and a neural network classifier (N) run on the TAC KBP 2016 document corpus.

Chapter 4

The price of debiasing automatic metrics in natural language evaluation

For evaluating generation systems, automatic metrics such as BLEU, which cost nothing to run, have been used to score the output generated by a system based on its similarity with the reference answer provided with the test collection. Unfortunately, these automatic metrics have been shown to correlate poorly with human judgment, leading to systematic bias against certain model improvements. On the other hand, averaging human judgments, the unbiased gold standard, is often too expensive to run.

In the previous chapter, we tackled a similar problem in tasks with finite incompleteness by combining data collected across multiple systems and thus amortizing costs. In this chapter, we'll look at a much harder problem, tackling *infinite incompleteness* in evaluating generation systems: in the infinite incompleteness setting, few predictions made by any two systems exactly overlap and thus the techniques of the previous chapter do not apply.

Instead, we use control variates to combine automatic metrics with human evaluation to obtain an unbiased estimator with lower cost than human evaluation alone. In practice, however, we obtain only a 7–13% cost reduction on evaluating summarization and open-response question answering systems. We then prove that our estimator is optimal: there is no unbiased estimator with lower cost. Our theory further highlights the two fundamental bottlenecks—the automatic metric and the prompt shown to human evaluators—both of which need to be improved to obtain greater cost savings.

4.1 Introduction

In recent years, there has been an increasing interest in tasks that require generating natural language, including abstractive summarization (Nallapati et al., 2016), open-response question answering (Nguyen et al., 2016; Kočisky et al., 2017), image captioning (Lin et al., 2014), and open-domain dialogue (Lowe et al., 2017b). Unfortunately, the evaluation of these systems remains a thorny issue because of the diversity of possible correct responses. As the gold standard of performing human evaluation is often too expensive, there has been a large effort developing automatic metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin and Rey, 2004), METEOR (Lavie and Denkowski, 2009; Denkowski and Lavie, 2014) and CiDER (Vedantam et al., 2015). However, these have shown to be biased, correlating poorly with human metrics across different datasets and systems (Liu et al., 2016b; Novikova et al., 2017).

Can we combine automatic metrics and human evaluation to obtain an *unbiased* estimate at *lower cost* than human evaluation alone? In this chapter, we propose a simple estimator based on control variates (Ripley, 2009), where we average differences between human judgments and automatic metrics rather than averaging the human judgments alone. Provided the two are correlated, our estimator will have lower variance and thus reduce cost.

We prove that our estimator is *optimal* in the sense that no unbiased estimator using the same automatic metric can have lower variance. We also analyze its data efficiency (equivalently, cost savings)—the factor reduction in number of human judgments needed to obtain the same accuracy versus naive human evaluation—and show that it depends solely on two factors: (a) the annotator variance (which is a function of the human evaluation prompt) and (b) the correlation between human judgments and the automatic metric. This factorization allows us to calculate typical and best-case data efficiencies and accordingly refine the evaluation prompt or automatic metric.

Finally, we evaluate our estimator on state-of-the-art systems from two tasks, summarization on the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016) and open-response question answering on the MS MARCOv1.0 dataset (Nguyen et al., 2016). To study our estimators offline, we preemptively collected 10,000 human judgments which cover several tasks and systems.[1] As predicted by the theory, we find that the data efficiency depends not only on the correlation between the human and automatic metrics, but also on the evaluation prompt. If the automatic metric had perfect correlation, our data efficiency would be around 3, while if we had noiseless human judgments, our data efficiency would be about 1.5. In reality, the reduction in cost we obtained was only about 10%, suggesting that improvements in both automatic metric and evaluation prompt are needed. As one case study in improving the latter, we show that, when compared to a Likert survey, measuring the amount of post-editing needed to fix a generated sentence reduced the annotator variance by three-fold.

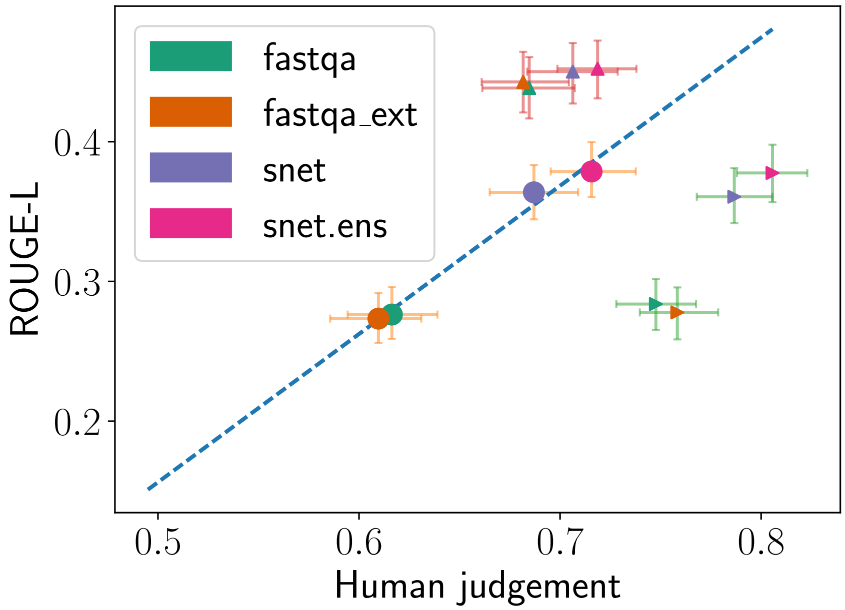
4.2 Bias in automatic evaluation

It is well understood that current automatic metrics tend to correlate poorly with human judgment at the instance-level. For example, Novikova et al. (2017) report correlations less than 0.3 for a large suite of word-based and grammar-based evaluation methods on a generation task. Similarly, Liu et al. (2016b) find correlations less than 0.35 for automatic metrics on a dialog generation task in one domain, but find correlations with the same metric dropped significantly to less than 0.16 when used in another domain. Still, somewhat surprisingly, several automatic metrics have been found to have high *system-level* correlations (Novikova et al., 2017). What, then, are the implications of having a low instance-level correlation?

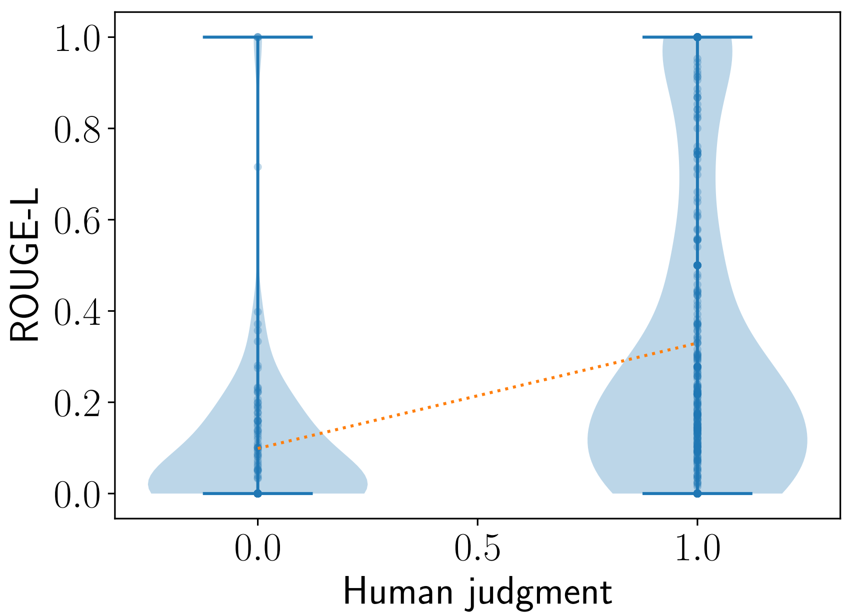
As a case study, consider the task of open-response question answering: here, a system receives a human-generated question and must *generate* an answer from some given context, e.g. a document or several webpages. We collected the responses of several systems on the MS MARCOv1 dataset (Nguyen et al., 2016) and crowdsourced human evaluations of the system output (see Section 4.4 for details).

The instance-level correlation (Figure 4.1b) is only $\rho = 0.31$. A closer look at the instance-level correlation reveals that while ROUGE is able to correctly assign low scores to bad examples (lower left), it is bad at judging good examples and often assigns them low

[1] An anonymized version of this data and the annotation interfaces used can be found at https://bit.ly/price-of-debiasing.



(a) System-level correlation on the MS MARCO task



(b) Instance-level correlation for the fastqa system

Figure 4.1: (a) At a system-level, automatic metrics (ROUGE-L) and human judgment correlate well, but (b) the instance-level correlation plot (where each point is a system prediction) shows that the instance-level correlation is quite low ($\rho = 0.31$). As a consequence, if we try to locally improve systems to produce better answers ($\triangleright$ in (a)), they do not significantly improve ROUGE scores and vice versa ($\triangle$).

Question and reference answer	**System answer (System;** Corr **/ ROUGE-L)**
Examples where system is correct and ROUGE-L > 0.5 *(*19.6% *or 285 of 1455 unique responses)*	
Q. what is anti-mullerian hormone **A.** Anti-Mullerian Hormone (AMH) is a protein hormone produced by granulosa cells (cells lining the egg sacs or follicles) within the ovary.	it is a protein hormone produced by granulosa cells (cells lining the egg sacs or follicles) within the ovary. ($\mathrm{snet.ens}$; ✓ / 0.86)
Examples where system is incorrect and ROUGE-L > 0.5 *(*1.3% *or 19 of 1455 unique responses)*	
Q. at what gestational age can you feel a fetus move **A.** 37 to 41 weeks *(incorrect reference answer)*	37 to 41 weeks (fastqa, $\mathrm{fastqa.ext}$; $\times$ / 1.0)
Examples where system is correct and ROUGE-L < 0.5 *(*56.0% *or 815 of 1455 unique responses)*	
Q. what is the definition of onomatopoeia **A.** It is defined as a word, which imitates the natural sounds of a thing.	the naming of a thing or action by a vocal imitation of the sound associated with it (as buzz, hiss). (fastqa; ✓ / 0.23)
Examples where system is incorrect and ROUGE-L < 0.5 *(*23.1% *or 336 of 1455 unique responses)*	
Q. what kind root stem does a dandelion have **A.** Fibrous roots and hollow stem.	vitamin a, vitamin c, vitamin d and vitamin b complex, as well as zinc, iron and potassium. (snet, $\mathrm{snet.ens}$; $\times$ / 0.09)

Table 4.1: Examples highlighting the different modes in which the automatic metric and human judgments may agree or disagree on the MS MARCO task. Human annotators rated answer correctness ($\mathrm{AnyCorrect}$) and the automatic metric used is ROUGE-L (higher is better). A majority of responses from systems were actually correct but poorly scored according to ROUGE-L.

Reference summary	**System summary (System;** Edit **/ VecSim)**
Examples where system Edit < 0.3 *and VecSim* > 0.5 *(*53.9% *or 1078 of 2000 responses)*	
Bhullar is set to sign a ■-day contract with the Kings. The ■-year-old will become the NBA's first player of Indian descent. Bhullar will be on the roster when the Kings host New Orleans Pelicans.	~~Bhullar and~~**The** Kings are signing Bhullar to a ■-day contract. The ■-year-old will be on the roster on friday when David Wear's ■-season contract expires thursday. Bhullar is set to become the NBA's first player of Indian descent. (ml; 0.13 / 0.82)
Examples where system Edit > 0.3 *and VecSim* > 0.5 *(*18.0% *or 360 of 2000 responses)*	
The Direct Marketing Commission probing B2C Data and Data Bubble. Investigating whether they breached rules on the sale of private data. Chief commissioner described allegations made about firms as 'serious'.	~~■ Data obtained by the Mail's marketing commission said it would probe both companies over claims that they had breached the rules on the sale of private data.~~ The FSA said it would probe both companies over claims they had breached the rules on the sale of private data. (se2seq; 1.00 / 0.72)
Examples where system Edit < 0.3 *and VecSim* < 0.5 *(*14.5% *or 290 of 2000 responses)*	
Death toll rises to more than ■. Pemba Tamang, ■, shows no apparent signs of serious injury after rescue. Americans special forces helicopter ■, including ■ Americans, to safety.	~~Six of~~ **Despite** Nepal's tragedy, life triumphed in Kathmandu's hard-hit neighborhoods. Rescuers pulled an 15-year-old from the rubble of a multistory residential building. He was wearing a New York shirt and a blue neck brace. (pointer; 0.04 / 0.27)
Examples where system Edit > 0.3 *and VecSim* < 0.5 *(*13.6% *or 272 of 2000 responses)*	
"Mad Men's" final seven episodes begin airing April ■. The show has never had high ratings but is considered one of the great TV series. It's unknown what will happen to characters, but we can always guess.	'~~This's~~ "Mad Men" is the end of a series of an era', ~~This~~ **he** says. Stores have created fashion lines inspired by the show.~~"The Sopranos". The in ■ the Kent State shootings in may ■ or Richard Nixonś ■ re-election..~~ (ml+rl; 0.95 / 0.24)

Table 4.2: Examples highlighting the different modes in which the automatic metric and human judgments may agree or disagree on the CNN/Daily Mail task. Human judgment scores used are post-edit distance (Edit) (lower is better) and the automatic metric used is sentence vector similarity with the reference (higher is better). A significant number of examples which are scored highly by VecSim are poorly rated by humans, and likewise many examples scored poorly by VecSim are highly rated by humans.

ROUGE scores (lower right)—see Table 4.1 and Table 4.2 for examples. This observation agrees with a finding reported in Novikova et al. (2017) that automatic metrics correlate better with human judgments on bad examples than average or good examples.

Thus, as Figure 4.1(a) shows, we can improve low-scoring ROUGE examples without improving their human judgment ($\triangle$) and vice versa ($\triangleright$). Indeed, Conroy and Dang (2008) report that summarization systems were optimized for ROUGE during the DUC challenge (Dang, 2006) until they were indistinguishable from the ROUGE scores of human-generated summaries, but the systems had hardly improved on human evaluation. Hill-climbing on ROUGE can also lead to a system that does worse on human scores, e.g. in machine translation (Wu et al., 2016). Conversely, genuine quality improvements might not be reflected in improvements in ROUGE. This bias also appears in pool-based evaluation for knowledge base population (Chaganty et al., 2017a). Thus the problems with automatic metrics clearly motivate the need for human evaluation, but can we still use the automatic metrics somehow to save costs?

4.3 Statistical estimation for unbiased evaluation

We will now formalize the problem of combining human evaluation with an automatic metric. Let $\mathcal{X}$ be a set of inputs (e.g., articles), and let S be the *system* (e.g. for summarization), which takes $x \in \mathcal{X}$ and returns output $S(x)$ (e.g. a summary). Let $\mathcal{Z} = \{(x, S(x)) : x \in \mathcal{X}\}$ be the set of system predictions. Let $Y(z)$ be the random variable representing the human judgment according to some evaluation prompt (e.g. grammaticality or correctness), and define $f(z) = \mathbb{E}[Y(z)]$ to be the (unknown) *human metric* corresponding to averaging over an infinite number of human judgments. Our goal is to estimate the average across all examples:

$$\mu \stackrel{\text{def}}{=} \mathbb{E}_z[f(z)] = \frac{1}{|\mathcal{Z}|} \sum_{z \in \mathcal{Z}} f(z) \tag{4.1}$$

with as few queries to Y as possible.

Let g be an automatic metric (e.g. ROUGE), which maps z to a real number. We assume evaluating $g(z)$ is free. The central question is how to use g in conjunction with calls to

Y to produce an unbiased estimate $\hat{\mu}$ (that is, $\mathbb{E}[\hat{\mu}] = \mu$). In this section, we will construct a simple estimator based on control variates (Ripley, 2009), and prove that it is minimax optimal.

4.3.1 Sample mean

We warm up with the most basic unbiased estimate, the sample mean. We sample $z^{(1)}, \dots, z^{(n)}$ independently with replacement from $\mathcal{Z}$. Then, we sample each human judgment $y^{(i)} = Y(z^{(i)})$ independently.[2] Define the estimator to be $\hat{\mu}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}$. Note that $\hat{\mu}_{\text{mean}}$ is unbiased ($\mathbb{E}[\hat{\mu}_{\text{mean}}] = \mu$).

We can define $\sigma_f^2 \stackrel{\text{def}}{=} \text{Var}(f(z))$ as the variance of the human metric and $\sigma_a^2 \stackrel{\text{def}}{=} \mathbb{E}_z[\text{Var}(Y(z))]$ as the variance of human judgment averaged over $\mathcal{Z}$. By the law of total variance, the variance of our estimator is

$$\text{Var}(\hat{\mu}_{\text{mean}}) = \frac{1}{n}(\sigma_f^2 + \sigma_a^2). \tag{4.2}$$

4.3.2 Control variates estimator

Now let us see how an automatic metric g can reduce variance. If there is no annotator variance ($\sigma_a^2 = 0$) so that $Y(z) = f(z)$, we should expect the variance of $f(z) - g(z)$ to be lower than the variance of $f(z)$, assuming g is correlated with f—see Figure 4.2 for an illustration.

The actual control variates estimator needs to handle noisy $Y(z)$ (i.e. $\sigma_a^2 > 0$) and guard against a $g(z)$ with low correlation. Let us standardize g to have zero mean and unit variance, because we have assumed it is free to evaluate. As before, let $z^{(1)}, \dots, z^{(n)}$ be independent samples from $\mathcal{Z}$ and draw $y^{(i)} = Y(z^{(i)})$ independently as well. We define the *control variates estimator* as

$$\hat{\mu}_{\text{cv}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} - \alpha g(z^{(i)}), \tag{4.3}$$

[2]Note that this independence assumption isn't quite true in practice since we do not control who annotates our data.

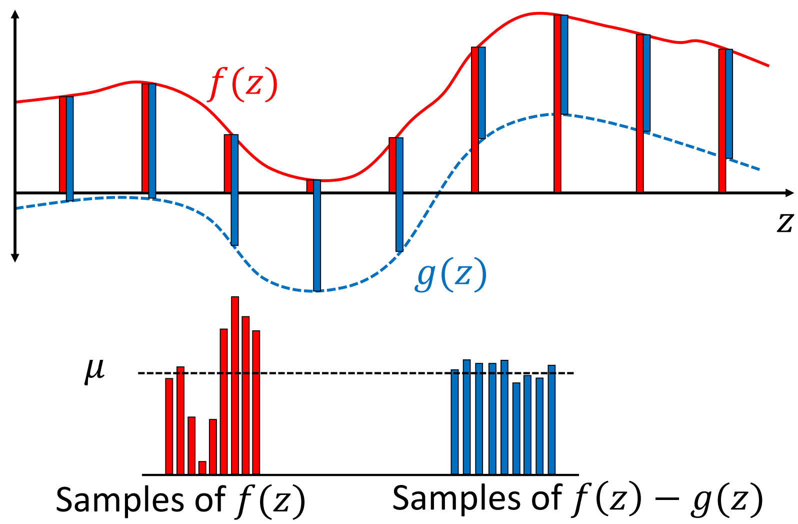


Figure 4.2: The samples from $f(z)$ have a higher variance than the samples from $f(z) - g(z)$ but the same mean. This is the key idea behind using control variates to reduce variance.

where

$$\alpha \stackrel{\text{def}}{=} \operatorname{Cov}(f(z), g(z)). \tag{4.4}$$

Intuitively, we have averaged over $y^{(i)}$ to handle the noise introduced by $Y(z)$, and scaled $g(z)$ to prevent an uncorrelated automatic metric from introducing too much noise.

An important quantity governing the quality of an automatic metric g is the correlation between $f(z)$ and $g(z)$ (recall that g has unit variance):

$$\rho \stackrel{\text{def}}{=} \frac{\alpha}{\sigma_f}. \tag{4.5}$$

We can show that among all distributions with fixed σ_f^2, σ_a^2, and α (equivalently ρ), this estimator is minimax optimal, i.e. it has the least variance among all unbiased estimators:

Theorem 1. *Among all unbiased estimators that are functions of $y^{(i)}$ and $g(z^{(i)})$, and for all distributions with a given σ_f^2, σ_a^2, and α,*

$$\operatorname{Var}(\hat{\mu}_{cv}) = \frac{1}{n}(\sigma_f^2(1-\rho^2) + \sigma_a^2), \tag{4.6}$$

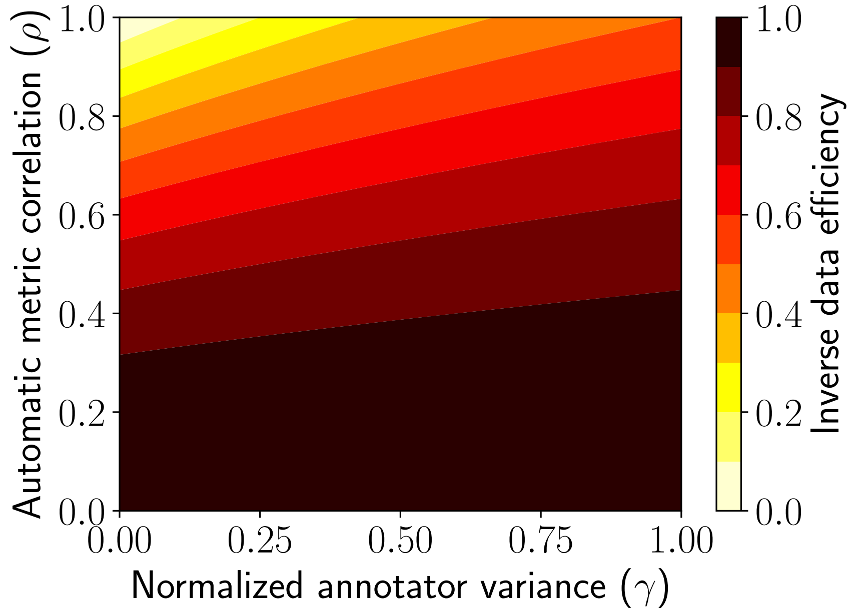


Figure 4.3: Inverse data efficiency for various values of γ and ρ. We need both low γ and high ρ to obtain significant gains.

and no other estimator has a lower worst-case variance.

Comparing the variances of the two estimators ((4.2) and (4.6)), we define the *data efficiency* as the ratio of the variances:

$$\mathrm{DE} \stackrel{\text{def}}{=} \frac{\mathrm{Var}(\hat{\mu}_{\text{mean}})}{\mathrm{Var}(\hat{\mu}_{\text{cv}})} = \frac{1+\gamma}{1-\rho^2+\gamma}, \tag{4.7}$$

where $\gamma \stackrel{\text{def}}{=} \sigma_a^2/\sigma_f^2$ is the normalized annotator variance. Data efficiency is the key quantity in this chapter: it is the multiplicative reduction in the number of samples required when using the control variates estimator $\hat{\mu}_{\text{cv}}$ versus the sample mean $\hat{\mu}_{\text{mean}}$. Figure 4.3 shows the inverse data efficiency contours as a function of the correlation ρ and γ.

When there is no correlation between human and automatic metrics ($\rho = 0$), the data efficiency is naturally 1 (no gain). In order to achieve a data efficiency of 2 (half the labeling cost), we need $|\rho| \geq \sqrt{2}/2 \approx 0.707$. Interestingly, even for an automatic metric with perfect correlation ($\rho = 1$), the data efficiency is still capped by $\frac{1+\gamma}{\gamma}$: unless $\gamma \to 0$ the data efficiency cannot increase unboundedly. Intuitively, even if we knew that $\rho = 1$, $f(z)$ would be undetermined up to a constant additive shift and just estimating the shift would incur a variance of $\frac{1}{n}\sigma_a^2$.

4.3.3 Using the control variates estimator

The control variates estimator can be easily integrated into an existing evaluation: we run human evaluation on a random sample of system outputs, automatic evaluation on all the system outputs, and plug in these results into Algorithm 1.

It is vital that we are able to evaluate the automatic metric on a significantly larger set of examples than those with human evaluations to reliably normalize $g(z)$: without these additional examples, it be can shown that the optimal minimax estimator for μ is simply the naive estimate $\hat{\mu}_{\text{mean}}$. Intuitively, this is because estimating the mean of $g(z)$ incurs an equally large variance as estimating μ. In other words, $g(z)$ is only useful if we have additional information about g beyond the samples $\{z^{(i)}\}$.

Algorithm 1 shows the estimator. In practice, we do not know $\alpha = \text{Cov}(f(z), g(z))$, so we use a plug-in estimate $\hat{\alpha}$ in line 3 to compute the estimate $\widetilde{\mu}$ in line 4. We note that estimating α from data does introduce a $O(1/n)$ bias, but when compared to the standard deviation which decays as $\Theta(1/\sqrt{n})$, this bias quickly goes to 0.

Proposition 1. *The estimator $\widetilde{\mu}$ in Algorithm 1 has $O(1/n)$ bias.*

Algorithm 1 Control variates estimator

1: **Input:** n human evaluations $y^{(i)}$ on system outputs $z^{(i)}$, *normalized* automatic metric g
2: $\overline{y} = \frac{1}{n}\sum_i y^{(i)}$
3: $\hat{\alpha} = \frac{1}{n}\sum_i (y^{(i)} - \overline{y})g(z^{(i)})$
4: $\widetilde{\mu} = \frac{1}{n}\sum_i y^{(i)} - \hat{\alpha}g(z^{(i)})$
5: **return** $\widetilde{\mu}$

An additional question that arises when applying Algorithm 1 is figuring out how many samples n to use. Given a target variance, the number of samples can be estimated using (4.6) with conservative estimates of σ_f^2, σ_a^2 and ρ. Alternatively, our estimator can be

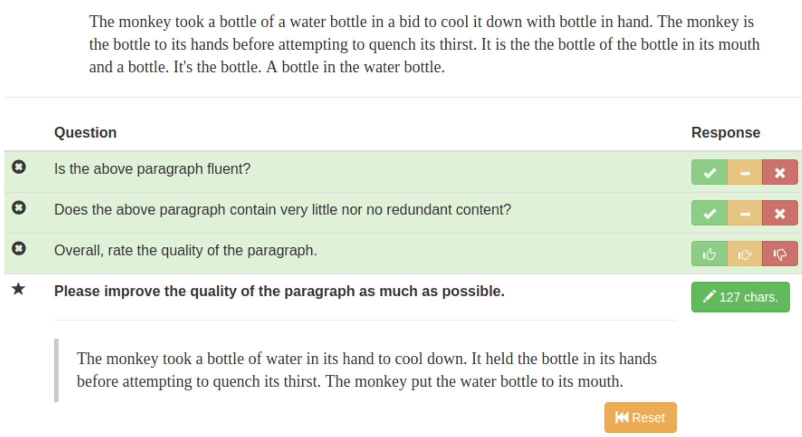
Task	Eval.	σ_a^2	σ_f^2	$\gamma = \frac{\sigma_a^2}{\sigma_f^2}$
CDM	Fluency	0.32	0.26	1.23
CDM	Redund.	0.26	0.43	0.61
CDM	Overall	0.28	0.28	1.00
CDM	**Edit**	**0.07**	**0.18**	**0.36**
MS MARCO	AnyCorr.	0.14	0.15	0.95
MS MARCO	AvgCorr.	0.12	0.13	0.91

Table 4.3: A summary of the key statistics, human metric variance (σ_f^2) and annotator variance (σ_a^2) for different datasets, CNN/Daily Mail (CDM) and MS MARCO in our evaluation benchmark. We observe that the relative variance (γ) is fairly high for most evaluation prompts, upper bounding the data efficiency on these tasks. A notable exception is the Edit prompt wherein systems are compared on the number of post-edits required to improve their quality.

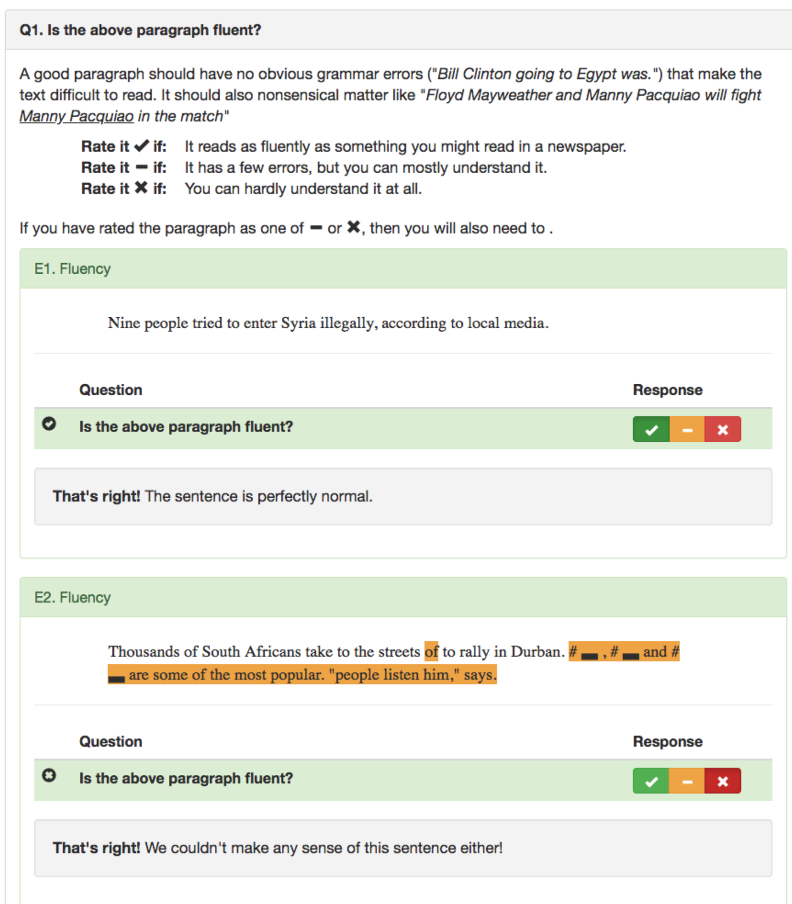
combined with a dynamic stopping rule (Mnih et al., 2008) to stop data collection once we reach a target confidence interval.

4.3.4 Discussion of assumptions

We will soon see that empirical instantiations of γ and ρ lead to rather underwhelming data efficiencies in practice. In light of our optimality result, does this mean there is no hope for gains? Let us probe our assumptions. We assumed that the human judgments are uncorrelated across different system outputs; it is possible that a more accurate model of human annotators (e.g. Passonneau and Carpenter (2014)) could offer improvements. Perhaps with additional information about $g(z)$ such as calibrated confidence estimates, we would be able to sample more adaptively. Of course the most direct routes to improvement involve increasing the correlation of g with human judgments and reducing annotator variance, which we will discuss more later.

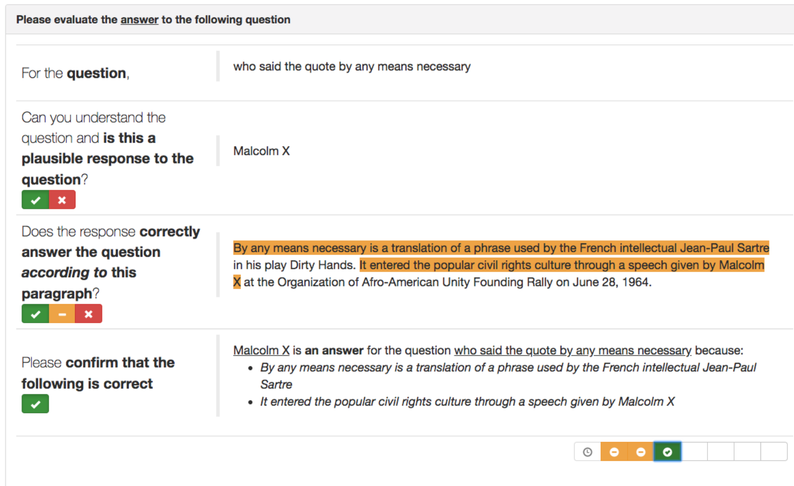


(a)

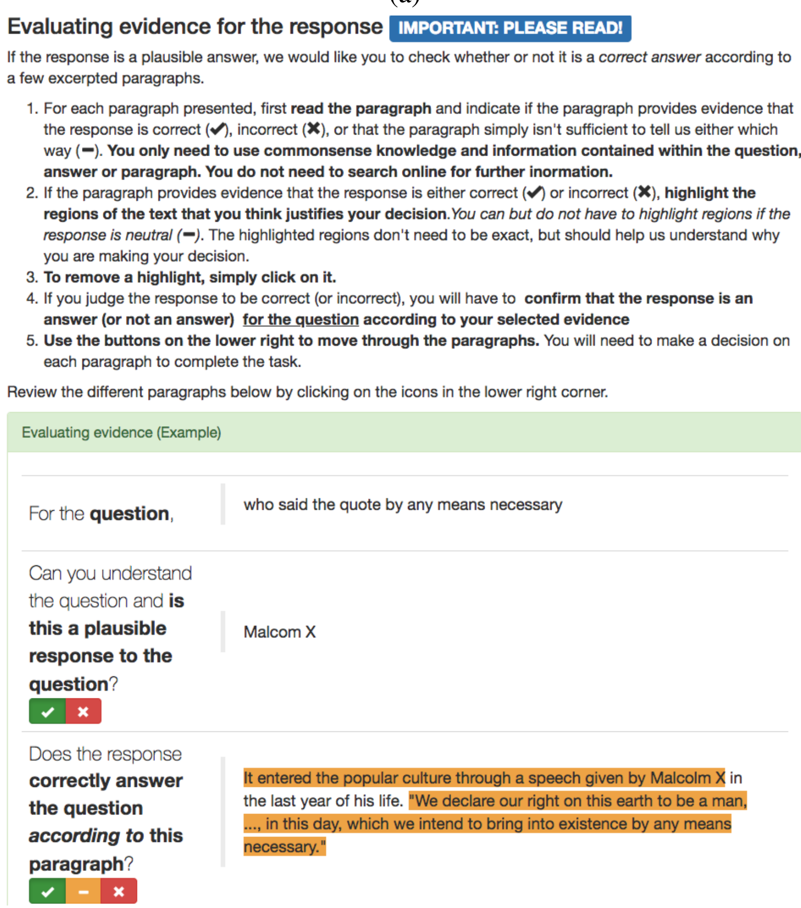


(b)

Figure 4.4: Screenshot of the (a) interface and (b) instructions used by crowdworkers for the language quality evaluation task on the CNN/Daily Mail dataset.



(a)



(b)

Figure 4.5: Screenshot of the (a) interface and (b) instructions used by crowdworkers for the answer correctness evaluation task on the MS MARCO dataset.

4.4 Tasks and datasets

In order to compare different approaches to evaluating systems, we first collected human judgments for the output of several automatic summarization and open-response question answering systems using Amazon Mechanical Turk. In this section, we'll briefly describe how we collected this data.

4.4.1 Evaluating language quality in automatic summarization

In automatic summarization, systems must generate a short (on average two or three sentence) summary of an article: for our study, we chose articles from the CNN/Daily Mail (CDM) dataset (Hermann et al., 2015; Nallapati et al., 2016) which come paired with reference summaries in the form of story highlights. We focus on the *language quality* of summaries and leave evaluating content selection to future work.

For each summary, we collected human judgments on a scale from 1–3 (Figure 4.4) for fluency, (lack of) redundancy, and overall quality of the summary using guidelines from the DUC summarization challenge (Dang, 2006). As an alternate human metric, we also asked workers to post-edit the system's summary to improve its quality, similar to the post-editing step in MT evaluations (Snover et al., 2006). Obtaining judgments costs about $0.15 per summary and this cost rises to about $0.40 per summary for post-editing.

Interface design choices. We found that using a five-level Likert scale increased annotator variance as annotators relative to a three-level Likert scale. Annotators were provided specific cues to calibrate their Likert ratings through a tutorial and were reminded of these cues through tooltips on the rating buttons (see Figure 4.4b for an example). If the annotators rated a summary as lacking along any facet, they were then forced to perform post-edits to "improve [its] quality as much as possible". We found that forcing annotators to provide post-edits on examples significantly decreased the annotator variance even on the Likert ratings.

Following the recommendations of Liu et al. (2016a), we forced annotators to complete an interactive tutorial containing 10 questions each before beginning the task (Figure 4.4b). The tutorial provided guidelines and examples on how to rate each facet (fluency, redundancy and overall quality) and tested whether they were able to identify and correct language errors using the post-editing interface. The tutorial took about 5–6 minutes to complete and annotators were paid a one-time bonus of $0.75 on completion.

We initially included additional questions to assess focus, coherency and referential clarity adapted from the DUC evaluation guidelines (Dang, 2006), but found that annotators were unable to reliably identify these errors in the short summaries. We also experimented with asking annotators to highlight language errors in the text to justify their ratings, but again found that annotators were unable to localize these errors reliably.

Quality control measures. We initially attempted to use attention-check examples for the Likert rating questions, but found that the ratings on these examples were themselves quite subjective and hence were not a reliable signal to reject work. Instead, we found that requiring post-edits to summaries significantly reduced spam. Additionally, we rejected annotators who took too little time to complete the task, had very low agreement rates on the Likert questions or had edits that were consistently shorter than 5 characters to prevent spam.

Overview of data collected. We collected judgments on the summaries generated by the $\mathrm{seq2seq}$ and $\mathrm{pointer}$ models of See et al. (2017), the ml and $\mathrm{ml{+}rl}$ models of Paulus et al. (2018), and the reference summaries.[3] Before presenting the summaries to human annotators, we performed some minimal post-processing: we true-cased and de-tokenized the output of $\mathrm{seq2seq}$ and $\mathrm{pointer}$ using Stanford CoreNLP (Manning et al., 2014) and replaced "unknown" tokens in each system with a special symbol (■).

4.4.2 Evaluating answer correctness.

Next, we look at evaluating the correctness of system outputs in question answering using the MS MARCO question answering dataset (Nguyen et al., 2016). Here, each system is provided with a question and up to 10 paragraphs of context. The system generates open-response answers that do not need to be tied to a span in any paragraph.

[3] All system output was obtained from the original authors through private communication.

We first ask annotators to judge if the output is even plausible for the question, and if yes, ask them identify if it is correct according to each context paragraph. We found that requiring annotators to highlight regions in the text that support their decision substantially improved the quality of the output without increasing costs. Annotations cost $0.40 per system response.[4]

While our goal is to evaluate the correctness of the provided answer, we found that there are often answers which may be correct or incorrect depending on the context. For example, the question "what is a pothole" is typically understood to refer to a hole in a roadway, but also refers to a geological feature (Figure 4.5). This is reflected when annotators mark one context paragraph to support the given answer but mark another to contradict it. We evaluated systems based on both the average correctness (AvgCorrect) of their answers across all paragraphs as well as whether their answer is correct according to any paragraph (AnyCorrect).

Interface design choices. We found that some of the questions in the MS MARCO dataset were extremely ambiguous (e.g. "metatarsal what causes") and some system responses were implausible (e.g "monogenic bone diseases", for the question "what genes cause osteoporosis"). In these cases, annotators expressed confusion if they were forced to judge if the response was correct or incorrect. We resolved this confusion by first asking annotators if the question made sense and if system response was even plausible.

In early pilots, we found that annotators often rated a paragraph that correctly answered the question but was unrelated to the system response to be "correct". We were able to resolve this problem by asking annotators to double-check their work (see the last question in Figure 4.5a for an example).

Once again, we forced annotators to complete an interactive tutorial containing eight questions each before beginning the task (Figure 4.5b). The tutorial also took about 5–6 minutes to complete and annotators were paid a one-time bonus of $0.75 on completion.

[4]This cost could be significantly reduced if systems also specify which passage they used to generate the answer.

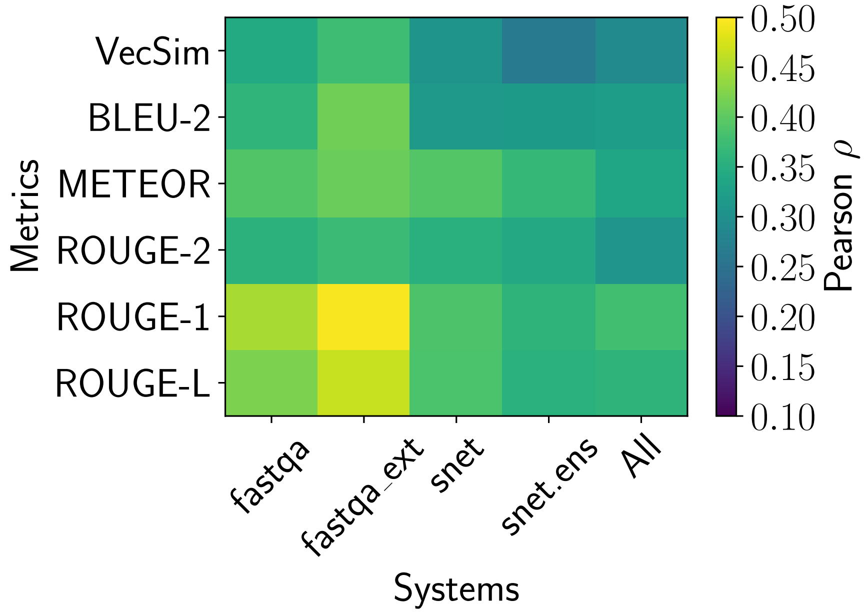
Quality control measures. We found that requiring annotators to provide justification spans significantly spam. Additionally, we rejected annotators who took too little time to complete the task or had very low agreement rates on the answer correctness.

Overview of data collected. We collected annotations on the systems generated by the fastqa and $\mathrm{fastqa\_ext}$ from Weissenborn et al. (2017) and the snet and $\mathrm{snet.ens}$(emble) models from Tan et al. (2018), along with reference answers. The answers generated by the systems were used without any post-processing. Surprisingly, we found that the correctness of the reference answers (according to the AnyCorrect metric) was only 73.5%, only 2% above that of the leading system ($\mathrm{snet.ens}$). We manually inspected 30 reference answers which were annotated incorrectly and found that of those, about 95% were indeed incorrect. However, 62% are actually answerable from some paragraph, indicating that the real ceiling performance on this dataset is around 90% and that there is still room for improvement on this task.

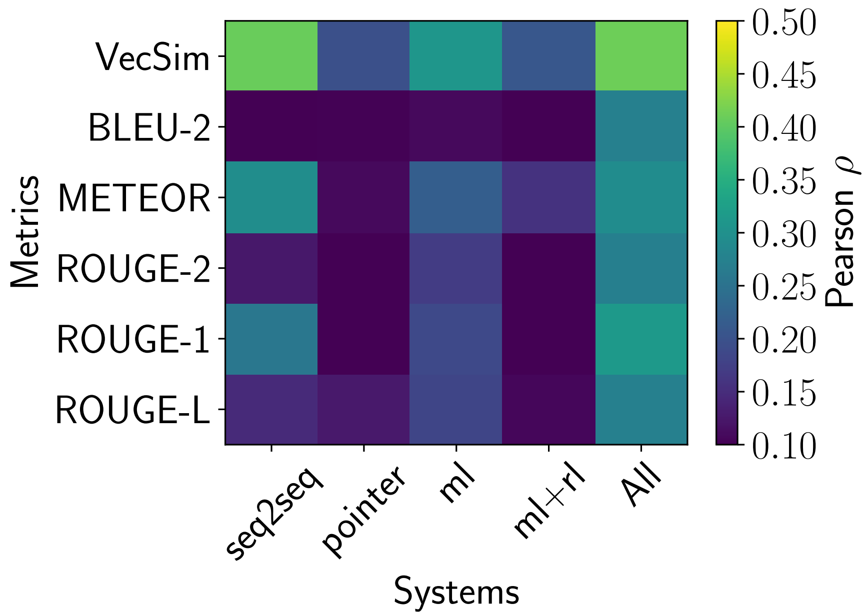
4.5 Experimental results

We are now ready to evaluate the performance of our control variates estimator proposed in Section 4.3 using the datasets presented in Section 4.4. Recall that our primary quantity of interest is *data efficiency*, the ratio of the number of human judgments required to estimate the overall human evaluation score for the control variates estimator versus the sample mean. We'll briefly review the automatic metrics used in our evaluation before analyzing the results.

Automatic metrics. We consider the following frequently used automatic word-overlap based metrics in our work: **BLEU** (Papineni et al., 2002), **ROUGE** (Lin and Rey, 2004) and **METEOR** (Lavie and Denkowski, 2009). Following Novikova et al. (2017) and Liu et al. (2016b), we also compared a vector-based sentence-similarity using $\mathrm{sent2vec}$ (Pagliardini et al., 2017) to compare sentences (**VecSim**). Figure 4.6 shows how each of these metrics is correlated with human judgment for the systems being evaluated. Unsurprisingly, the correlation varies considerably across systems, with token-based metrics correlating more



(a) MS MARCO with the AnyCorrect prompt



(b) CNN/Daily Mail with the Edit prompt

Figure 4.6: Correlations of different automatic metrics on the MS MARCO and CNN/Daily Mail tasks. Certain systems are more correlated with certain automatic metrics than others, but overall the correlation is low to moderate for most systems and metrics.

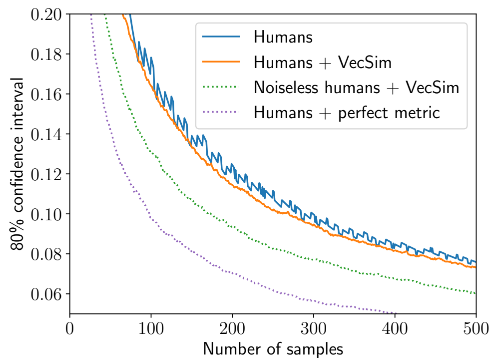
strongly for systems that are more extractive in nature (fastqa and $\mathrm{fastqa\_ext}$).

Results. In Section 4.3 we proved that the control variates estimator is not only unbiased but also has the least variance among other unbiased estimators. Figure 4.7 plots the width of the 80% confidence interval, estimated using bootstrap, measured as a function of the number of samples collected for different tasks and prompts. As expected, the control variates estimator reduces the width of the confidence interval. We measure data efficiency by the averaging of the ratio of squared confidence intervals between the human baseline and control variates estimates. We observe that the data efficiency depends on the task, prompt and system, ranging from about 1.08 (a 7% cost reduction) to 1.15 (a 13% cost reduction) using current automatic metrics.

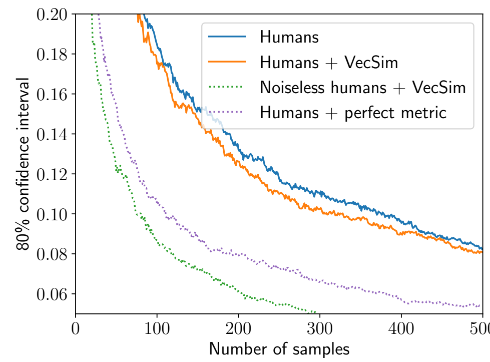
As we showed in Section 4.3, further gains are fundamentally limited by the quality of the evaluation prompts and automatic metrics. Figures 4.7a and 4.7b show how improving the quality of the evaluation prompt from a Likert-scale prompt for quality ($\mathrm{Overall}$) to using post-editing (Edit) noticeably decreases variance and hence allows better automatic metrics to increase data efficiency. Likewise, Figure 4.7c shows how using a better automatic metric (ROUGE-L instead of VecSim) also reduces variance.

Figure 4.7 also shows the conjectured confidence intervals if we were able to eliminate noise in human judgments (noiseless humans) or have a automatic metric that correlated perfectly with average human judgment (perfect metric). In particular, we use the mean of all (2–3) humans on each z for the perfect $g(z)$ and use the mean of all humans on each z for the "noiseless" $Y(z)$.

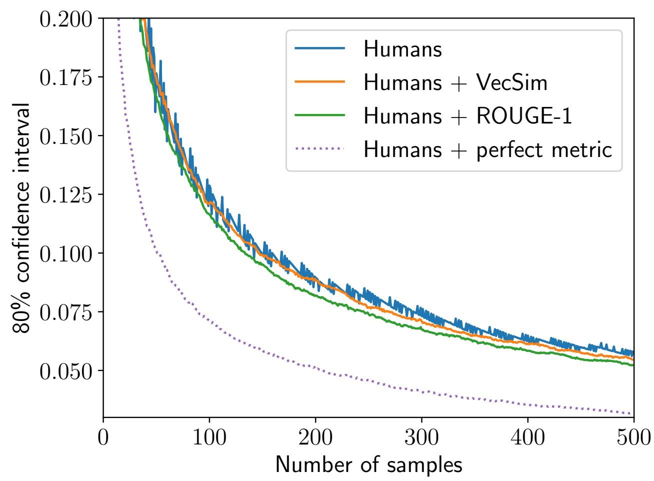
In both cases, we are able to significantly increase data efficiency (i.e. decrease estimator variance). With zero annotator variance and using existing automatic metrics, the data efficiency ranges from 1.42 to 1.69. With automatic metrics with perfect correlation and current variance of human judgments, it ranges from 2.38 to 7.25. Thus, we conclude that it is important not only to improve our automatic metrics but also the evaluation prompts we use during human evaluation.



(a) seq2seq on CNN/Daily Mail using the Overall



(b) seq2seq on CNN/Daily Mail using Edit



(c) fastqa_ext on MS MARCO using AnyCorrect

Figure 4.7: 80% bootstrap confidence interval length as a function of the number of human judgments used when evaluating the indicated systems on their respective datasets and prompts. (a) We see a modest reduction in variance (and hence cost) relative to human evaluation by using the VecSim automatic metric with the proposed control variates estimator to estimate Overall scores on the CNN/Daily Mail task; the data efficiency (DE) is 1.06. (b) By improving the evaluation prompt to use Edits instead, it is possible to further reduce variance relative to humans (DE is 1.15). (c) Another way to reduce variance relative to humans is to improve the automatic metric evaluation; here using ROUGE-1 instead of VecSim improves the DE from 1.03 to 1.16.

4.6 Related work

In this chapter, we focus on using existing automatic metrics to decrease the cost of human evaluations. There has been much work on improving the quality of automatic metrics. In particular, there is interest in learning models (Lowe et al., 2017a; Dusek et al., 2017) that are able to optimize for improved correlations with human judgment. However, in our experience, we have found that these learned automatic metrics have trouble generalizing to different systems. The framework we provide allows us to safely incorporate such models into evaluation, exploiting them when their correlation is high but also not introducing bias when it is low.

Our key technical tool is control variates, a standard statistical technique used to reduce the variance of Monte Carlo estimates (Ripley, 2009). The technique has also been used in machine learning and reinforcement learning to lower variance estimates of gradients (Greensmith et al., 2004; Paisley et al., 2012; Ranganath et al., 2014). To the best of our knowledge, we are the first to apply this technique in the context of language evaluation.

Our work also highlights the importance of human evaluation. Chaganty et al. (2017a) identified a similar problem of systematic bias in evaluation metrics in the setting of knowledge base population and also propose statistical estimators that relies on human evaluation to correct bias. Unfortunately, their technique relies on having a structured output (relation triples) that are shared between systems and does not apply to evaluating natural language generation. In a similar vein, Chang et al. (2017) dynamically collect human feedback to learn better dialog policies.

4.7 Discussion

Prior work has shown that existing automatic metrics have poor instance-level correlation with mean human judgment and that they score many good quality responses poorly. As a result, the evaluation is systematically biased against genuine system improvements that would lead to higher human evaluation scores but not improve automatic metrics. In this chapter, we have explored using an automatic metric to decrease the cost of human evaluation without introducing bias. In practice, we find that with current automatic metrics and evaluation prompts data efficiencies are only 1.08–1.15 (7–13% cost reduction). Our theory shows that further improvements are only possible by improving the correlation of the automatic metric and reducing the annotator variance of the evaluation prompt. As an example of how evaluation prompts could be improved, we found that using post-edits of summarizes decreased normalized annotator variance by a factor of three relative to using a Likert scale survey. It should be noted that changing the evaluation prompt also changes the underlying ground truth $f(z)$: it is up to us to find a prompt that still captures the essence of what we want to measure.

Without making stronger assumptions, the control variates estimator we proposed outlines the limitations of unbiased estimation. Where do we go from here? Certainly, we can try to improve the automatic metric (which is potentially as difficult as solving the task) and brainstorming alternative ways of soliciting evaluation (which has been less explored). Alternatively, we could give up on measuring absolute scores, and seek instead to find techniques to stably rank methods and thus improve them.

Finally, it is interesting to contrast the limitations on variance reductions we have proved in this chapter with the more encouraging results of Chapter 3. While both methods guarantee unbiasedness, in Chapter 3 we were able to exploit exact matches between the output of different systems to reuse annotations and hence amortize costs. On the other hand, for the tasks studied in this chapter it is rare that two systems generate the exact same output: as a result the only way to share information between systems is through the automatic metric which we have seen to be a bottleneck in reducing variance. Our findings apply not only to other text generation tasks such as machine translation or image captioning but in any setting with infinite incompleteness.

Chapter 5

On-the-Job Learning with Bayesian Decision Theory

Having seen the power of introducing human annotations to properly evaluate systems in the last two chapters, we now explore ways in which we can use human feedback to address *incompleteness in the training data*. The crux of the problem is that our training data, particularly when it is small, is often unable to adequately provide the model enough information to generalize to new types of unseen test instances. At the same time, it is hard to know apriori exactly which examples are necessary to add to the training data to improve the system's performance.

In this chapter, we propose a new learning paradigm, "on-the-job" learning, which uses the system to identify input that it is uncertain about *at test-time* and request for on-demand human feedback to address its uncertainty. In this manner, the model indirectly identifies incompleteness in the training data through examples in the test data that it is uncertain about. It then resolves the incompleteness by collecting annotations on those examples. The net effect is a system that can maintain high accuracy irrespective of how much training data it has. However, when doing so in practice, there are additional constraints, annotation cost and response latency, that apply. Our key idea here is to cast the problem as a stochastic game based on Bayesian decision theory, which allows us to balance latency, cost, and accuracy objectives in a principled way. We test our approach on three different tasks, named-entity recognition, sentiment classification and image classification, and show that we are able to reduce annotation costs by an order of magnitude relative to full human annotation by using on-the-job learning without any loss in accuracy.

5.1 Introduction

There are two roads to an accurate AI system today: (i) gather a huge amount of labeled training data (Deng et al., 2009) and do supervised learning (Krizhevsky et al., 2012); or (ii) use crowdsourcing to directly perform the task (Bernstein et al., 2010; Kokkalis et al., 2013). However, both solutions require non-trivial amounts of time and money. In many situations, one wishes to build a new system — e.g., to do Twitter information extraction (Li et al., 2012) to aid in disaster relief efforts or monitor public opinion — but one simply lacks the resources to follow either the pure ML or pure crowdsourcing road.

In this chapter, we propose a framework called *on-the-job learning* (formalizing and extending ideas first implemented in Lasecki et al. (2013)), in which we produce high quality results from the start without requiring a trained model. When a new input arrives, the system can choose to asynchronously query the crowd on *parts* of the input it is uncertain about (e.g. query about the label of a single token in a sentence). After collecting enough evidence the system makes a prediction. The goal is to maintain high accuracy by initially using the crowd as a crutch, but gradually becoming more self-sufficient as the model improves. Online learning (Cesa-Bianchi and Lugosi, 2006) and online active learning (Helmbold and Panizza, 1997; Sculley, 2007; Chu et al., 2011) are different in that they do not actively seek new information *prior* to making a prediction, and cannot maintain high accuracy independent of the number of data instances seen so far. Active classification (Gao and Koller, 2011), like us, strategically seeks information (by querying a subset of labels) prior to prediction, but it is based on a static policy, whereas we improve the model during test time based on observed data.

To determine which queries to make, we model on-the-job learning as a stochastic game based on a CRF prediction model. We use Bayesian decision theory to tradeoff latency, cost, and accuracy in a principled manner. Our framework naturally gives rise to intuitive strategies: To achieve high accuracy, we should ask for redundant labels to offset the noisy responses. To achieve low latency, we should issue queries in parallel, whereas if

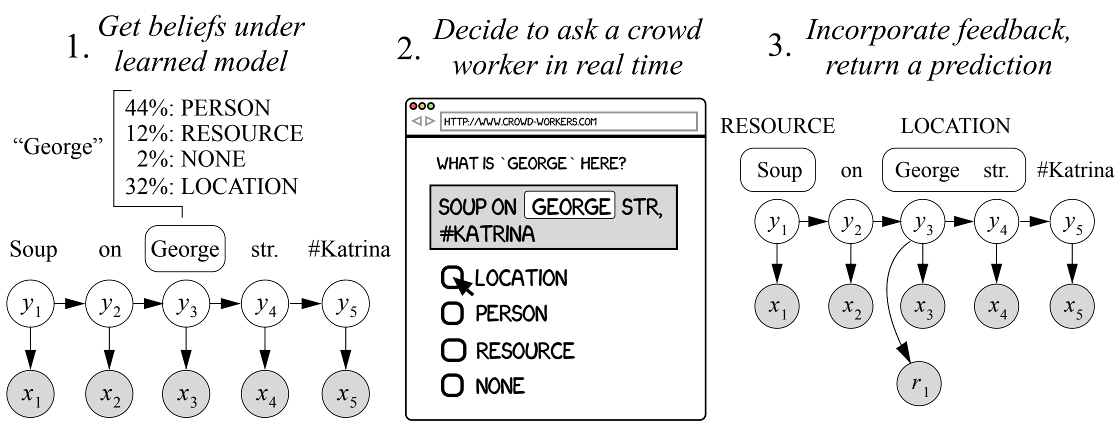


Figure 5.1: Named entity recognition on tweets in on-the-job learning.

latency is unimportant, we should issue queries sequentially in order to be more adaptive. Computing the optimal policy is intractable, so we develop an approximation based on Monte Carlo tree search (Kocsis and Szepesvári, 2006) and progressive widening to reason about continuous time (Coulom, 2007).

We implemented and evaluated our system on three different tasks: named-entity recognition, sentiment classification, and image classification. On the NER task we obtained more than an order of magnitude reduction in cost compared to full human annotation, while boosting performance relative to the expert provided labels. We also achieve a 8% F1 improvement over having a single human label the whole set, and a 28% F1 improvement over online learning. An open-source implementation of our system, dubbed LENSE for "Learning from Expensive Noisy Slow Experts" is publicly available.[1]

5.2 Problem formulation

Consider a structured prediction problem from input $\mathbf{x} = (x_1, \ldots, x_n)$ to output $\mathbf{y} = (y_1, \ldots, y_n)$. For example, for named-entity recognition (NER) on tweets, $\mathbf{x}$ is a sequence of words in the tweet (e.g., "*on George str.*") and $\mathbf{y}$ is the corresponding sequence of labels (e.g., NONE LOCATION LOCATION). The full set of labels of PERSON, LOCATION,

[1] http://www.github.com/keenon/lense

RESOURCE, and NONE.

In the *on-the-job learning* setting, inputs arrive in a stream. On each input $\mathbf{x}$, we make zero or more queries $q_1, q_2, \dots$ on the crowd to obtain labels (potentially more than once) for any positions in $\mathbf{x}$. The responses $r_1, r_2, \dots$ come back asynchronously, which are incorporated into our current prediction model p_θ. Figure 5.3 shows one possible outcome: We query positions $q_1 = 2$ ("*George*") and $q_2 = 3$ ("*str.*"). The first query returns $r_1 =$ LOCATION, upon which we make another query on the the same position $q_3 = 3$ ("*George*"), and so on. When we have sufficient confidence about the entire output, we return the most likely prediction $\hat{\mathbf{y}}$ under the model. Each query q_i is issued at time s_i and the response comes back at time t_i. Assume that each query costs m cents. Our goal is to choose queries to maximize accuracy, minimize latency and cost.

We make several remarks about this setting: First, we must make a prediction $\hat{\mathbf{y}}$ on each input $\mathbf{x}$ in the stream, unlike in active learning, where we are only interested in the pool or stream of examples for the purposes of building a good model. Second, the responses are used to update the prediction model, like in online learning. This allows the number of queries needed (and thus cost and latency) to decrease over time without compromising accuracy.

5.3 Model

We model on-the-job learning as a stochastic game with two players: the system and the crowd. The game starts with the system receiving input $\mathbf{x}$ and ends when the system turns in a set of labels $\mathbf{y} = (y_1, \dots, y_n)$. During the system's turn, the system may choose a query action $q \in \{1, \dots, n\}$ to ask the crowd to label y_q. The system may also choose the wait action ($q = \varnothing_W$) to wait for the crowd to respond to a pending query or the return action ($q = \varnothing_R$) to terminate the game and return its prediction given responses received thus far. The system can make as many queries in a row (i.e. simultaneously) as it wants, before deciding to wait or turn in.[2] When the wait action is chosen, the turn switches to the crowd, which provides a response r to one pending query, and advances the game clock by the time taken for the crowd to respond. The turn then immediately reverts back to the

[2] This rules out the possibility of launching a query midway through waiting for the next response. However, we feel like this is a reasonable limitation that significantly simplifies the search space.

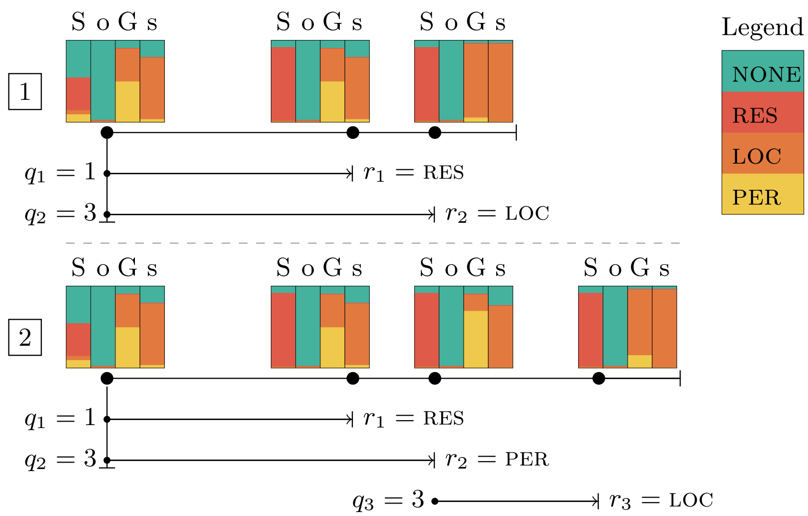


Figure 5.2: Example behavior while running structure prediction on the tweet "Soup on George str." The bar graphs represent the marginals over the labels for each token (indicated by the first character) at different points in time. The two timelines show how the system updates its confidence over labels based on the crowd's responses. The system continues to issue queries until it has sufficient confidence on its labels. See the paragraph on behavior in Section 5.3 for more information.

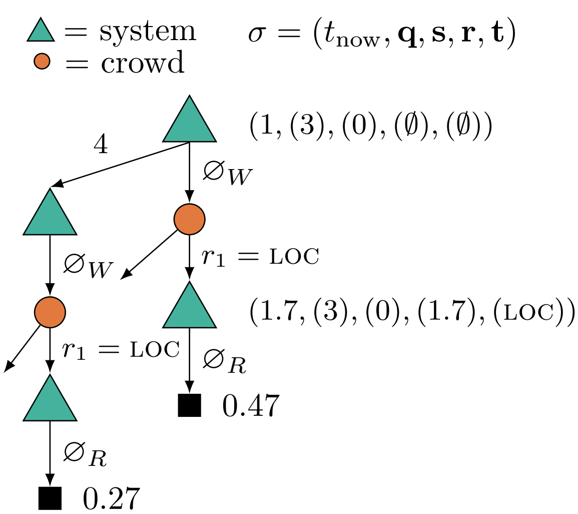


Figure 5.3: Example behavior while running structure prediction on the tweet "Soup on George str." A partial game tree constructed by the system when deciding which action to take in the state $\sigma = (1, (3), (0), (\emptyset), (\emptyset))$, i.e. the query $q_1 = 3$ has already been issued and the system must decide whether to issue another query or wait for a response to q_1.

system. When the game ends (the system chooses the return action), the system evaluates a utility that depends on the accuracy of its prediction, the number of queries issued and the total time taken. The system should choose query and wait actions to maximize the utility of the prediction eventually returned.

In the rest of this section, we describe the details of the game tree, our choice of utility and specify models for crowd responses, followed by a brief exploration of behavior admitted by our model.

Game tree. Let us now formalize the game tree in terms of its states, actions, transitions and rewards; see Figure 5.3 for an example. The *game state* $\sigma = (t_{\text{now}}, \mathbf{q}, \mathbf{s}, \mathbf{r}, \mathbf{t})$ consists of the current time t_{now}, the actions $\mathbf{q} = (q_1, \ldots, q_{k-1})$ that have been issued at times $\mathbf{s} = (s_1, \ldots, s_{k-1})$ and the responses $\mathbf{r} = (r_1, \ldots, r_{k-1})$ that have been received at times $\mathbf{t} = (t_1, \ldots, t_{k-1})$. Let $r_j = \emptyset$ and $t_j = \emptyset$ iff q_j is not a query action or its responses have not been received by time t_{now}.

During the system's turn, when the system chooses an action q_k, the state is updated to $\sigma' = (t_{\text{now}}, \mathbf{q}', \mathbf{s}', \mathbf{r}', \mathbf{t}')$, where $\mathbf{q}' = (q_1, \ldots, q_k)$, $\mathbf{s}' = (s_1, \ldots, s_{k-1}, t_{\text{now}})$, $\mathbf{r}' = (r_1, \ldots, r_{k-1}, \emptyset)$ and $\mathbf{t}' = (t_1, \ldots, t_{k-1}, \emptyset)$. If $q_k \in \{1, \ldots n\}$, then the system chooses another action from the new state σ'. If $q_k = \varnothing_W$, the crowd makes a stochastic move from σ'. Finally, if $q_k = \varnothing_R$, the game ends, and the system returns its best estimate of the labels using the responses it has received and obtains a utility $U(\sigma)$ (defined later).

Let $F = \{1 \leq j \leq k-1 | q_j \neq \varnothing_W \wedge r_j = \emptyset\}$ be the set of *in-flight* requests. During the crowd's turn (i.e. after the system chooses $\varnothing_W$), the next response from the crowd, $j^* \in F$, is chosen: $j^* = \arg\min_{j \in F} t'_j$ where t'_j is sampled from the *response-time model*, $t'_j \sim p_{\text{T}}(t'_j | s_j, t'_j > t_{\text{now}})$, for each $j \in F$. Finally, a response is sampled using a response model, $r'_{j^*} \sim p(r'_{j^*} | \mathbf{x}, \mathbf{r})$, and the state is updated to $\sigma' = (t_{j^*}, \mathbf{q}, \mathbf{s}, \mathbf{r}', \mathbf{t}')$, where $\mathbf{r}' = (r_1, \ldots, r'_{j^*}, \ldots, r_k)$ and $\mathbf{t}' = (t_1, \ldots, t'_{j^*}, \ldots, t_k)$.

Utility. Under Bayesian decision theory, the *optimal choice* for an action in state $\sigma = (t_{\text{now}}, \mathbf{q}, \mathbf{r}, \mathbf{s}, \mathbf{t})$ is the one that attains the maximum expected utility (i.e. value) for the game starting at σ. Recall that the system can return at any time, at which point it receives a utility that trades off two things: The first is the accuracy of the MAP estimate according to the model's best guess of $\mathbf{y}$ incorporating all responses received by time τ. The second is the cost of making queries: a (monetary) cost w_{M} per query made and penalty of w_{T} per unit of time taken. Formally, we define the utility to be:

$$U(\sigma) \stackrel{\text{def}}{=} \text{ExpAcc}(p(\mathbf{y}|\mathbf{x},\mathbf{q},\mathbf{s},\mathbf{r},\mathbf{t})) - (n_{\text{Q}} w_{\text{M}} + t_{\text{now}} w_{\text{T}}), \tag{5.1}$$

$$\text{ExpAcc}(p) = \mathbb{E}_{p(\mathbf{y})}[\text{Accuracy}(\arg\max_{\mathbf{y}'} p(\mathbf{y}'))], \tag{5.2}$$

where $n_{\text{Q}} = |\{j | q_j \in \{1, \dots, n\}|$ is the number of queries made, $p(\mathbf{y}|\mathbf{x},\mathbf{q},\mathbf{s},\mathbf{r},\mathbf{t})$ is a prediction model that incorporates the crowd's responses.

The utility of wait and return actions is computed by taking expectations over subsequent trajectories in the game tree. This is intractable to compute exactly, so we propose an approximate algorithm in Section 5.4.

Environment model. The final component is a model of the environment (crowd). Given input $\mathbf{x}$ and queries $\mathbf{q} = (q_1, \dots, q_k)$ issued at times $\mathbf{s} = (s_1, \dots, s_k)$, we define a distribution over the output $\mathbf{y}$, responses $\mathbf{r} = (r_1, \dots, r_k)$ and response times $\mathbf{t} = (t_1, \dots, t_k)$ as follows:

$$p(\mathbf{y},\mathbf{r},\mathbf{t}|\mathbf{x},\mathbf{q},\mathbf{s}) \stackrel{\text{def}}{=} p(\mathbf{y}|\mathbf{x}) \prod_{i=1}^{k} p_{\text{R}}(r_i \mid y_{q_i}) p_{\text{T}}(t_i \mid s_i). \tag{5.3}$$

The three components are as follows: $p(\mathbf{y}|\mathbf{x})$ is the *prediction model* (e.g. a standard linear-chain CRF); $p_{\text{R}}(r|y_q)$ is the *response model* which describes the distribution of the crowd's response r for a given a query q when the true answer is y_q; and $p_{\text{T}}(t_i \mid s_i)$ specifies the latency of query q_i. The CRF model $p(\mathbf{y}|\mathbf{x})$ is learned based on all actual responses (not simulated ones) using AdaGrad. To model annotation errors, we set $p_{\text{R}}(r|y_q) = 0.7$ iff $r = y_q$,[3] and distribute the remaining probability for r uniformly. Given this full model, we can compute $p(r' \mid \mathbf{x}, \mathbf{r}, q)$ simply by marginalizing out $\mathbf{y}$ and $\mathbf{t}$ from (5.3). When conditioning on $\mathbf{r}$, we ignore responses that have not yet been received (i.e. when $r_j = \emptyset$ for some j).

[3] We found the humans we hired were roughly 70% accurate in our experiments

Behavior. Let's look at typical behavior that we expect the model and utility to capture. Figure 5.2 shows how the marginals over the labels change as the crowd provides responses for our running example, i.e. named entity recognition for the sentence "Soup on George str.". In the both timelines, the system issues queries on "Soup" and "George" because it is not confident about its predictions for these tokens. In the first timeline, the crowd correctly responds that "Soup" is a resource and that "George" is a location. Integrating these responses, the system is also more confident about its prediction on "str.", and turns in the correct sequence of labels. In the second timeline, a crowd worker makes an error and labels "George" to be a person. The system still has uncertainty on "George" and issues an additional query which receives a correct response, following which the system turns in the correct sequence of labels. While the answer is still correct, the system could have taken less time to respond by making an additional query on "George" at the very beginning.

5.4 Game playing

In Section 5.3 we modeled on-the-job learning as a stochastic game played between the system and the crowd. We now turn to the problem of actually finding a policy that maximizes the expected utility, which is, of course, intractable because of the large state space.

Our algorithm (Algorithm 2) combines ideas from Monte Carlo tree search (Kocsis and Szepesvári, 2006) to systematically explore the state space and progressive widening (Coulom, 2007) to deal with the challenge of continuous variables (time). Some intuition about the algorithm is provided below. When simulating the system's turn, the next state (and hence action) is chosen using the upper confidence tree (UCT) decision rule that trades off maximizing the value of the next state (exploitation) with the number of visits (exploration). The crowd's turn is simulated based on transitions defined in Section 5.3. To handle the unbounded fanout during the crowd's turn, we use progressive widening that maintains a current set of "active" or "explored" states, which is gradually grown with time. Let $N(\sigma)$ be the number of times a state has been visited, and $C(\sigma)$ be all successor states that the algorithm has sampled.

Algorithm 2 Approximating expected utility with MCTS and progressive widening

For all σ, $N(\sigma) \leftarrow 0$, $V(\sigma) \leftarrow 0$, $C(\sigma) \leftarrow [\,]$ ▷ Initialize visits, utility sum, and children
function MONTECARLOVALUE(state σ)
    increment $N(\sigma)$
    if system's turn **then**
        $\sigma' \leftarrow \arg\max_{\sigma'} \left\{ \frac{V(\sigma')}{N(\sigma')} + c\sqrt{\frac{\log N(\sigma)}{N(\sigma')}} \right\}$ ▷ Choose next state σ' using UCT
        $v \leftarrow$MONTECARLOVALUE(σ')
        $V(\sigma) \leftarrow V(\sigma) + v$ ▷ Record observed utility
        return v
    else if crowd's turn **then**
        if $\max(1, \sqrt{N(\sigma)}) \leq |C(\sigma)|$ **then** ▷ Restrict continuous samples using PW
            σ' is sampled from set of already visited $C(\sigma)$ based on (5.3)
        else
            σ' is drawn based on (5.3)
            $C(\sigma) \leftarrow C(\sigma) \cup \{[\sigma']\}$
        end if
        return MONTECARLOVALUE(σ')
    else if game terminated **then**
        return utility U of σ according to (5.1)
    end if
end function

5.5 Experiments

In this section, we empirically evaluate our approach on three tasks. While the on-the-job setting we propose is targeted at scenarios where there is no data to begin with, we use existing labeled datasets (Table 5.1) to have a gold standard.

Baselines. We evaluated the following four methods on each dataset:

1. **Human n-query**: The majority vote of n human crowd workers was used as a prediction.

2. **Online learning**: Uses a classifier that trains on the gold output for all examples seen so far and then returns the MLE as a prediction. This is the best possible offline system: it sees perfect information about all the data seen so far, but cannot query the crowd while making a prediction.

3. **Threshold baseline**: Uses the following heuristic: For each label, y_i, we ask for m queries such that $(1 - (()\, y_i|\mathbf{x})) \times 0.3^m \geq 0.98$. Instead of computing the expected marginals over the responses to queries in flight, we simply count the in-flight requests for a given variable, and reduces the uncertainty on that variable by a factor of 0.3. The system continues launching requests until the threshold (adjusted by number of queries in flight) is crossed. Predictions are made using MLE on the model given responses. The baseline does not reason about time and makes all its queries at the very beginning.

4. **LENSE:** Our full system as described in Section 5.3.

Implementation and crowdsourcing setup. We implemented the retainer model of Bernstein et al. (2011) on Amazon Mechanical Turk to create a "pool" of crowd workers that could respond to queries in real-time. The workers were given a short tutorial on each task before joining the pool to minimize systematic errors caused by misunderstanding the task.

[4] http://www.cnts.ua.ac.be/conll2003/ner/

[5] The original also includes a fifth tag for miscellaneous, however the definition for miscellaneos is complex, making it very difficult for non-expert crowd workers to provide accurate labels.

Dataset (Examples)	Task and notes	Features
NER (657)	We evaluate on the CoNLL-2003 NER task[4], a sequence labeling problem over English sentences. We only consider the four tags corresponding to persons, locations, organizations or none[5].	We used standard features (Finkel et al., 2005): the current word, current lemma, previous and next lemmas, lemmas in a window of size three to the left and right, word shape and word prefix and suffixes, as well as word embeddings.
Sentiment (1800)	We evaluate on a subset of the IMDB sentiment dataset (Maas et al., 2011) that consists of 2000 polar movie reviews; the goal is binary classification of documents into classes POS and NEG.	We used two feature sets, the first (UNIGRAMS) containing only word unigrams, and the second (RNN) that also contains sentence vector embeddings from Socher et al. (2013).
Face (1784)	We evaluate on a celebrity face classification task (Kumar et al., 2009). Each image must be labeled as one of the following four choices: Andersen Cooper, Daniel Craig, Scarlet Johansson or Miley Cyrus.	We used the last layer of a 11-layer AlexNet (Krizhevsky et al., 2012) trained on ImageNet as input feature embeddings, though we leave back-propagating into the net to future work.

Table 5.1: Datasets used in this chapter and number of examples we evaluate on.

We paid workers \$1.00 to join the retainer pool and an additional \$0.01 per query (for NER, since response times were much faster, we paid \$0.005 per query). Worker response times were generally in the range of 0.5–2 seconds for NER, 10–15 seconds for Sentiment, and 1–4 seconds for Faces.

When running experiments, we found that the results varied based on the current worker quality. To control for variance in worker quality across our evaluations of the different methods, we collected 5 worker responses and their delays on each label ahead of time[6]. During simulation we sample the worker responses and delays without replacement from this frozen pool of worker responses.

[6]These datasets are available in the code repository for this chapter

System	Delay/tok	Qs/tok	PER F_1	LOC F_1	ORG F_1	F_1
1-vote	467 ms	1.0	90.2	78.8	71.5	80.2
3-vote	750 ms	3.0	93.6	85.1	74.5	85.4
5-vote	1350 ms	5.0	**95.5**	87.7	78.7	87.3
Online	n/a	n/a	56.9	74.6	51.4	60.9
Threshold	414 ms	0.61	95.2	**89.8**	79.8	88.3
LENSE	**267 ms**	**0.45**	95.2	89.7	**81.7**	**88.8**

Table 5.2: Results on the NER task comparing latencies, queries per token (Qs/tok) and F_1.

System	Latency	Qs/ex	Acc.
1-vote	1216 ms	1.0	93.6
3-vote	1782 ms	3.0	99.1
5-vote	2103 ms	5.0	99.8
Online	n/a	n/a	79.9
Threshold	1680 ms	2.66	93.5
LENSE	1590 ms	2.37	99.2

Table 5.3: Results on the Face task comparing latencies, queries per token (Qs/tok) and accuracy.

Summary of results. Table 5.2, Table 5.3 and Table 5.4 summarize the performance of the methods on the three tasks. On all three datasets, we found that on-the-job learning outperforms machine and human-only comparisons on both quality and cost. On NER, we achieve an F_1 of 88.4% at more than an order of magnitude reduction on the cost of achieving comporable quality result using the 5-vote approach. On Sentiment and Faces, we reduce costs for a comparable accuracy by a factor of around 2. For the latter two tasks, both on-the-job learning methods perform less well than in NER. We suspect this is due to the presence of a dominant class ("none") in NER that the model can very quickly learn to expend almost no effort on. LENSE outperforms the threshold baseline, supporting the importance of Bayesian decision theory.

Figure 5.5 tracks the performance and cost of LENSE over time on the NER task. LENSE is not only able to consistently outperform other baselines, but the cost of the

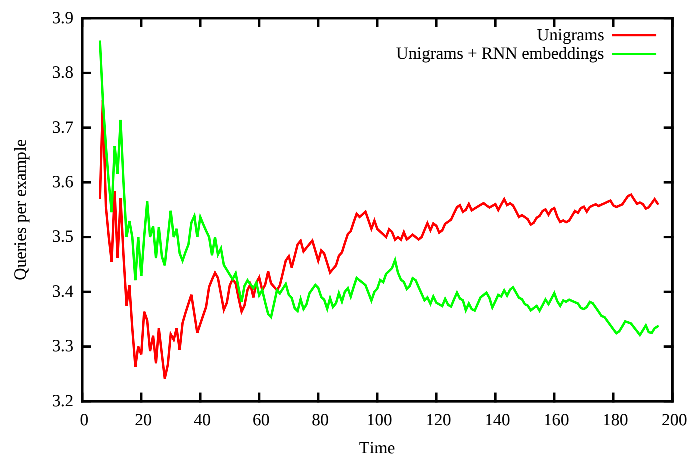


Figure 5.4: Queries per example for LENSE on Sentiment. With simple UNIGRAM features, the model quickly learns it does not have the capacity to answer confidently and must query the crowd. With more complex RNN features, the model learns to be more confident and queries the crowd less over time.

system steadily reduces over time. On the NER task, we find that LENSE is able to trade off time to produce more accurate results than the 1-vote baseline with fewer queries by waiting for responses before making another query.

While on-the-job learning allows us to deploy quickly and ensure good results, we would like to eventually operate without crowd supervision. Figure 5.4, we show the number of queries per example on Sentiment with two different features sets, UNIGRAMS and RNN (as described in Table 5.1). With simpler features (UNIGRAMS), the model saturates early and we will continue to need to query to the crowd to achieve our accuracy target (as specified by the loss function). On the other hand, using richer features (RNN) the model is able to learn from the crowd and the amount of supervision needed reduces over time. Note that even when the model capacity is limited, LENSE is able to guarantee a consistent, high level of performance.

Reproducibility. All code, data, and experiments for this chapter are available on CodaLab at https://www.codalab.org/worksheets/0x2ae89944846444539c2d08a0b7ff3f6f/.

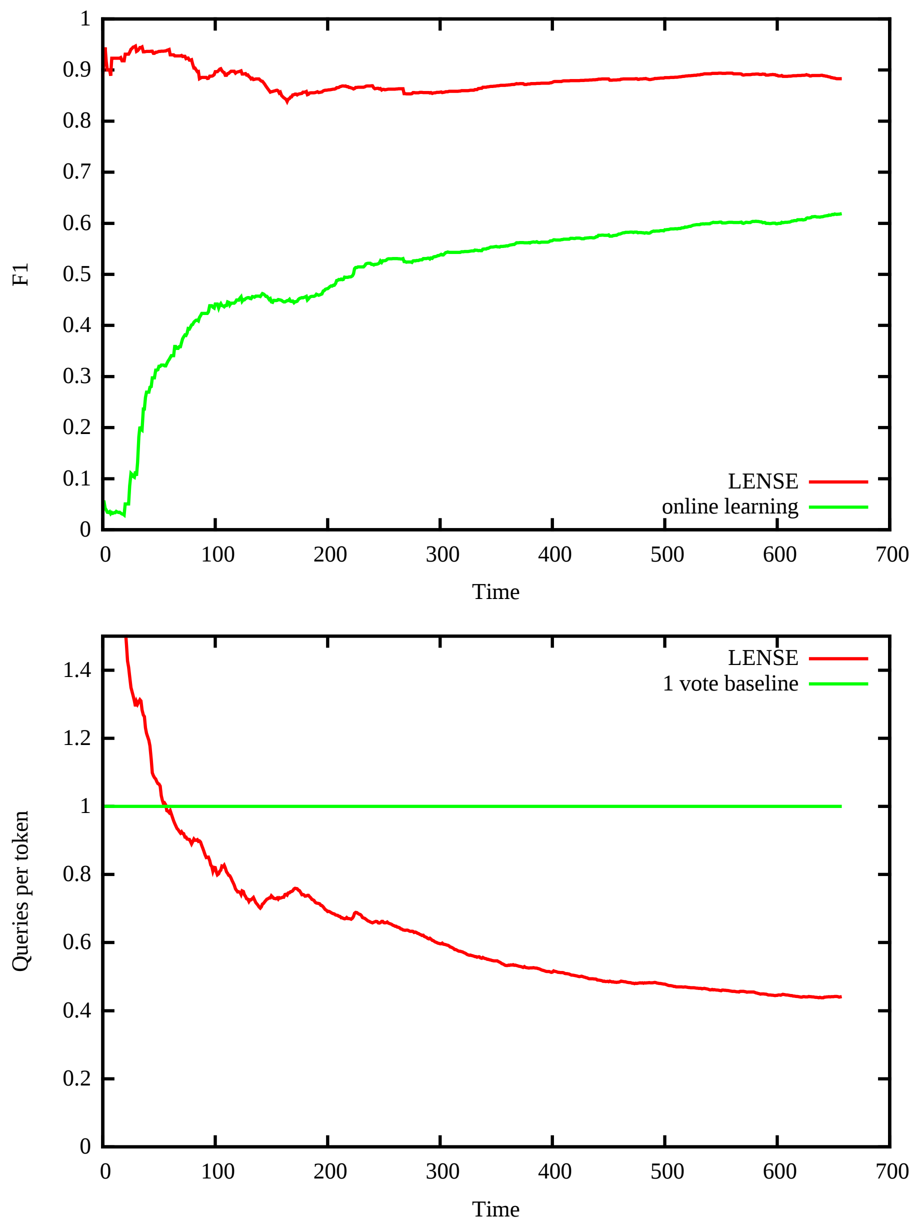


Figure 5.5: Comparing F_1 and queries per token on the NER task over time. The left graph compares LENSE to online learning (which cannot query humans at test time). This highlights that LENSE maintains high F_1 scores even with very small training set sizes, by falling back the crowd when it is unsure. The right graph compares query rate over time to 1-vote. This clearly shows that as the model learns, it needs to query the crowd less.

System	Latency	Qs/ex	Acc.
1-vote	6.6 s	1.00	89.2
3-vote	10.9 s	3.00	95.8
5-vote	13.5 s	5.00	98.7
	UNIGRAMS		
Online	n/a	n/a	78.1
Threshold	10.9 s	2.99	95.9
LENSE	11.7 s	3.48	98.6
	RNN		
Online	n/a	n/a	85.0
Threshold	11.0 s	2.85	96.0
LENSE	11.0 s	3.19	98.6

Table 5.4: Results on the Sentiment task comparing latency, queries per example and accuracy.

5.6 Related Work

On-the-job learning draws ideas from many areas: online learning, active learning, active classification, crowdsourcing, and structured prediction.

Online learning. The fundamental premise of online learning is that algorithms should improve with time, and there is a rich body of work in this area (Cesa-Bianchi and Lugosi, 2006). In our setting, algorithms not only improve over time, but maintain high accuracy from the beginning, whereas regret bounds only achieve this asymptotically.

Active learning. Active learning (see Settles (2010) for a survey) algorithms strategically select most informative examples to build a classifier. Online active learning (Helmbold and Panizza, 1997; Sculley, 2007; Chu et al., 2011) performs active learning in the online setting. Several authors have also considered using crowd workers as a noisy oracle e.g. (Donmez and Carbonell, 2008; Golovin et al., 2010). It differs from our setup in that it assumes that labels can only be observed *after* classification, which makes it nearly impossible to maintain high accuracy in the beginning.

Active classification. Active classification Greiner et al. (2002); Chai et al. (2004); Esmeir and Markovitch (2007) asks what are the *most informative features* to measure at test time. Existing active classification algorithms rely on having a fully labeled dataset which is used to learn a static policy for when certain features should be queried, which does not change at test time. On-the-job learning differs from active classification in two respects: true labels are *never* observed, and our system improves itself at test time by learning a stronger model. A notable exception is Legion:AR Lasecki et al. (2013), which like us operates in on-the-job learning setting to for real-time activity classification. However, they do not explore the machine learning foundations associated with operating in this setting, which is the aim of this chapter.

Crowdsourcing. A burgenoning subset of the crowdsourcing community overlaps with machine learning. One example is *Flock* Cheng and Bernstein (2015), which first crowdsources the identification of features for an image classification task, and then asks the crowd to annotate these features so it can learn a decision tree. In another line of work, *TurKontrol* Dai et al. (2010) models individual crowd worker reliability to optimize the number of human votes needed to achieve confident consensus using a POMDP.

Structured prediction. An important aspect our prediction tasks is that the output is structured, which leads to a much richer setting for one-the-job learning. Since tags are correlated, the importance of a coherent framework for optimizing querying resources is increased. Making active partial observations on structures and has been explored in the measurements framework of Liang et al. (2009) and in the distant supervision setting Angeli et al. (2014).

5.7 Conclusion

We have introduced a new framework that learns from (noisy) crowds *on-the-job* to maintain high accuracy, and reducing cost significantly over time. The technical core of our approach is modeling the on-the-job setting as a stochastic game and using ideas from game playing to approximate the optimal policy. We have built a system, LENSE, which obtains significant cost reductions over a pure crowd approach and significant accuracy improvements over a pure ML approach.

More broadly, in this chapter we have shown how the uncertainty estimates of a statistical model can be used to identify and resolve *incompleteness in the training data* at *test time*. In contrast to the methods in Chapter 3 and Chapter 4, the approach we propose in this chapter is not unbiased, but rather integrates human feedback into a statistical model and uses its own uncertainty to evaluate the performance of the model. As a result, the model is robust to annotation errors, which posed a problem in Chapter 4, that strongly contradict the models own prior belief by requesting for more annotations. On the contrary, the method relies on the model being well-calibrated and its estimated accuracy can be significantly wrong if this condition is not met. In addition to being pragmatic, requesting humans to correct model predictions is also important when evaluating multi-turn interactive systems such as dialogue systems: an error early on the conversation limits our ability to evaluate performance later in the conversation. By using on-demand human feedback, we can correct such errors before they arise and thus increase the scope of evaluation.

Chapter 6

Conclusions

In the course of this thesis, we have presented several examples of an unmet assumption of the current evaluation paradigm for challenging information summarization tasks: that test collections are *complete*, in that they contain the universe of all possible answers, and accurately represent the instances found in practice. In Chapter 3, we showed how the finite incompleteness inherent in information extraction tasks, such as knowledge base population, introduces significant biases in our evaluation. In Chapter 4, we showed that when incompleteness is infinite, such as in text generation tasks, bias is pervasive and hard to eliminate. Finally, in Chapter 5, we showed how incompleteness in the training data makes it hard to accurately evaluate or train systems without human intervention.

The crux of the solutions we provided in this thesis was combining on-demand human annotations with statistical techniques. The human annotations, enabled by crowdsourcing platforms such at Amazon Mechanical Turk, allow us to effectively side-step the problem of incompleteness. We have shown that using appropriate statistical techniques allows us to reduce the costs of collecting these annotations, sometimes by an order of magnitude.

In Chapter 3, we proposed importance-reweighted estimators for two widely used metrics, *precision* and *recall* and applied them to the task of knowledge base population. The estimator overcomes limitations with existing importance-weighted estimators that allow it to apply previously collected annotations to new systems. For both these metrics, we have been able to successfully reduce variance by a factor of 3–4 by amortizing over multiple systems in the finite incompleteness setting. We also showed how the estimator can be integrated into an online evaluation service, providing not only unbiased evaluation scores, but also a quantitative error analysis.

In Chapter 4, we identified an *optimal* estimator to *debias* automatic metrics like BLEU or ROUGE which are popularly used in tasks with infinite incompleteness like open-response question answering or text summarization. We prove that the poor correlation between the automatic metrics and human judgment fundamentally lower bounds the number of annotations needed to correct the bias; in practice, we need almost as many human annotations to correct the bias as to conduct an complete human evaluation! Our results shed light on how we can improve the evaluation procedures for such challenging tasks.

In Chapter 5, we use a statistical model to identify when it is uncertain about its own prediction and use that information to request crowdworkers for annotations *at test time*. By casting the problem as a Bayesian decision problem, we are able to balance accuracy, cost and latency. On three different classification tasks, we find an order of magnitude reduction in annotation relative to the human-only baseline, and significantly better accuracy relative to a model trained on a static dataset.

In the rest of this chapter, we will briefly discuss some challenges that face evaluation in natural language processing before concluding with a discussion of the role of evaluation in NLP.

6.1 Challenges for evaluation

In this section, we'll briefly review a few open challenges for the evaluation of complex tasks like information summarization as they relate to the work in this thesis.

6.1.1 The cost and latency of evaluation

While utilizing human feedback is currently necessary to evaluate the natural language processing tasks presented here, doing so incurs both costs to pay human annotators and latency. In Chapter 5, we used ideas from game playing to strategically *trade off* error rates, annotation cost and latency. Further improvements require new ideas.

One approach to decrease the costs of annotations is to *gamify* the task to incentivize people to provide labels for their personal enjoyment. For example, the ESP game (von Ahn and Dabbish, 2004) was successfully able to collect thousands of image labels by asking two randomly paired people to try to label the image with the same word. Gamification has also been applied to language tasks like word sense disambiguation (Vannella et al., 2014) and coreference resolution (Poesio et al., 2013). For all it success, gamifying complex annotation tasks can be quite challenging, even more so if annotations are desired immediately.

The latency of acquiring human annotations can often be improved by parallelizing across multiple annotators, optimizing the annotation interface or batching similar tasks to decrease worker context switching. Krishna et al. (2016) combine all these ideas to show how images can be successfully labeled at human response times by tapping into the annotator's reflexes: once aggregated over several annotators, they are able to speed up annotation by a factor of 10 with only a small reduction in speed.

6.1.2 Train-test mismatch

The ultimate goal of quantitative evaluation is to get a reliable indicator for the performance of a system were it applied in practice. Unfortunately, the machine learning community as a whole is trying to reconcile serious problems related to generalization even on classification tasks. For example, even slight perturbations to the pixels of an image can wildly throw off a state-of-the-art image classification system that scores extremely well on *the test set* (Goodfellow et al., 2015; Carlini and Wagner, 2016, 2017). In fact, Recht et al. (2018) report a generalization gap even when the new test data is constructed in an identical (but independently collected) manner to the original training data! Clearly, there is a problem in how we have been evaluating our systems and measuring progress thus far. Test data that *should* be representative of real-world instances does not seem to be.

In the field of natural language processing, too, there is a renewed discussion of how the performance of current NLP systems drops when applied to the real world (Plank, 2016): for example, McClosky (2010) show a 10–20 point gap between constituency parsing in domain and out of domain and Foster et al. (2011) find a similar gap on POS tagging and dependency parsing. More recently, Jia and Liang (2017) find that even minor edits to the source text can throw off state of the art reading comprehension models.

The problem of resolving the mismatch between training sets and test sets has been extensively studied in the machine learning community. Some popular routes to try and address this generalization problem include *domain adaptation* (Plank, 2011) (making the train and test data look more similar) and *transfer learning* (Weiss et al., 2016) (measuring the speed with which a system trained on a task A can learn to perform a task B). As Plank (2016) argue, domain adaptation makes the unrealistic assumption that we know what the target domain is; on the other hand, transfer learning may allow us to developing better learning algorithms, it does little to tell us about the performance of a system on our desired task!

Another way to view the problem is one of incompleteness in the training data—it does not contain all the information necessary for the model to be able to generalize to the test set. As a result, on-demand human annotation can also help resolve the mismatch between train-test sets. We provide one possible solution in Chapter 5 by using on-demand human annotations when the model is uncertain to guarantee high accuracy while minimizing annotation costs.

6.2 How should we evaluate evaluation?

Before we conclude, let's take a step back to ask what the goals of evaluation are. There is no doubt that having a robust, quantitative evaluation provides a clear direction for members in the research community to improve their systems and thus do science. At the same time, we must be wary of oversimplifying what it means to solve a problem into a single number: all too often these numbers hinge on over-fitting to dataset specific artifacts (Gururangan et al., 2018). While no evaluation is perfect, it is our opinion that we should strive for evaluation that provides us with *actionable insights* that allow us to *improve our systems*. We hope that our work in Chapter 3 provides a positive example of how annotation feedback can be used not only to provide a number to benchmark systems with, but also to provide error analysis and reveal new opportunities for research.

Next, we would like to revisit the importance of *unbiasedness* in evaluation. In Chapter 3 and Chapter 4, we adhered to the statistical definition of unbiasedness: we required that our statistical estimates of performance always match human judgment for the same, irrespective of the system. While this is an appealing guarantee for an evaluation methodology to have, we showed in Chapter 4 that it places fundamental limitations on how affordable we can make routine on-demand human evaluation. It is likely that we will have to introduce appropriate inductive bias to make the evaluation of tasks such as open-response question answering or text summarization practical. Unlike the systematic biases in the evaluation methodology we began with, carefully introducing inductive bias relies on assumptions that we can check and ensure are (mostly) met by the systems we evaluate. In contrast, we showed in Chapter 4 that automatic evaluation methods such as BLEU or ROUGE correlate so poorly with human judgment that they are not only weak indicators of performance, but are also easily gamed. We strongly advise researchers to ensure that the automatic metrics used correlate with human judgment *for the systems being evaluated* before using them.

In closing, we hope that this thesis has contributed to improving the state of evaluation in natural language processing. Still, there is much work ahead to improve the evaluation of natural language processing systems: we must remain humble in recognizing the limitations of any evaluation. As the techniques we seek to evaluate evolve, so too must our evaluation methodology.

Appendix A

Supplementary material for Chapter 3

A.1 Theoretical proofs for the sampling procedures

Let's refresh notation from Section 3.4.

Let $\mathcal{X}$ be a universe of possible outputs (e.g. relation instances), $\mathcal{Y} \subseteq \mathcal{X}$ be an unknown subset of this universe corresponding to the correct elements in $\mathcal{X}$ and $X_1, \ldots X_m \subseteq \mathcal{X}$ be known subsets that correspond to the predicted output from m systems, and $Y_1, \ldots, Y_m$ be the intersection of $X_1, \ldots, X_m$ with $\mathcal{Y}$. Furthermore, let $\hat{X}_i$ be a mulit-set of n_i independent samples drawn from X_i with the distribution p_i, $\hat{Y}_i$ be the intersection of these sets with $\mathcal{Y}$, and $\hat{Y}_0$ be a sample drawn from $\mathcal{Y}$ according to an unknown distribution $p'(x)$.

We would like to evaluate precision, π_i, and recall, r_i:

$$\pi_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim X_i}[f(x)] \qquad\qquad r_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim \mathcal{Y}}[g_i(x)],$$

In this section, we'll provide proofs that show that the joint estimators proposed in Section 3.4 are indeed unbiased, and we will characterize their variance.

A.1.1 Estimating precision

In Section 3.4, we proposed the following estimator for π_i:

$$\hat{\pi}_i \stackrel{\text{def}}{=} \sum_{j=1}^{m} \frac{w_{ij}}{n_j} \sum_{x \in \hat{X}_j} \frac{p_i(x) f(x)}{q_i(x)},$$

where $q_i(x) = \sum_{j=1}^{m} w_{ij} p_j(x)$ and $w_{ij} \geq 0$ are mixture parameters such that $\sum_{j=1}^{m} w_{ij} = 1$ and $q_i(x) > 0$ wherever $p_i(x) > 0$.

Theorem 2 (Statistical properties of $\hat{\pi}_i$)**.** *$\hat{\pi}_i$ is an unbiased estimator of π_i and has a variance of:*

$$\operatorname{Var} \hat{\pi}_i = \sum_{j=1}^{m} \frac{w_j^2}{n_j} \mathbb{E}_{p_j} \left[\frac{p_i(x)^2 f(x)^2 - \pi_{ij} p_i(x) f(x) q_i(x)}{q_i(x)^2} \right],$$

where $\pi_{ij} \stackrel{\text{def}}{=} \mathbb{E}_{p_j} \left[\frac{p_i(x) f(x)}{q_i(x)} \right]$.

Proof. Let $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_m)$ which is drawn from the product distribution of $p_1 \times p_m$. By independence and the linearity of expectation,

$$\mathbb{E}_{\hat{X}} \left[\sum_{j=1}^{m} f(\hat{X}_j) \right] = \sum_{j=1}^{m} \mathbb{E}_{\hat{X}_j} [f(\hat{X}_j)].$$

First, let's show that $\hat{\pi}_i$ is unbiased:

$$\begin{aligned}
\mathbb{E}_{\hat{X}}[\hat{\pi}_i] &= \mathbb{E}_{\hat{X}}\left[\sum_{j=1}^{m}\frac{w_j}{n_j}\sum_{x\in\hat{X}_j}\frac{p_i(x)f(x)}{q_i(x)}\right] \\
&= \sum_{j=1}^{m}\frac{w_j}{n_j}\mathbb{E}_{\hat{X}_j}\left[\sum_{x\in\hat{X}_j}\frac{p_i(x)f(x)}{q_i(x)}\right] \\
&= \sum_{j=1}^{m}\frac{w_j}{n_j}n_j\mathbb{E}_{p_j}\left[\frac{p_i(x)f(x)}{q_i(x)}\right] \\
&= \sum_{j=1}^{m}w_j\sum_{x\in\mathcal{X}}p_j(x)\frac{p_i(x)f(x)}{q_i(x)} \\
&= \sum_{x\in\mathcal{X}}\sum_{j=1}^{m}w_jp_j(x)\frac{p_i(x)f(x)}{q_i(x)} \\
&= \sum_{x\in\mathcal{X}}q_i(x)\frac{p_i(x)f(x)}{q_i(x)} \\
&= \sum_{x\in\mathcal{X}}p_i(x)f(x) \\
&= \pi_i.
\end{aligned}$$

Now let's compute the variance.

$$\begin{aligned}
\operatorname{Var}\hat{\pi}_i &= \sum_{j=1}^{m}\frac{w_j^2}{n_j}\mathbb{E}_{p_j}\left[\frac{p_i(x)^2f(x)^2}{q_i(x)^2}\right] - \sum_{j=1}^{m}\frac{w_j^2}{n_j}\mathbb{E}_{p_j}\left[\frac{p_i(x)f(x)}{q_i(x)}\right]^2 \\
&= \sum_{j=1}^{m}\frac{w_j^2}{n_j}\mathbb{E}_{p_j}\left[\frac{p_i(x)^2f(x)^2}{q_i(x)^2} - \frac{\pi_{ij}p_i(x)f(x)}{q_i(x)}\right] \\
&= \sum_{j=1}^{m}\frac{w_j^2}{n_j}\mathbb{E}_{p_j}\left[\frac{p_i(x)^2f(x)^2 - \pi_{ij}p_i(x)f(x)q_i(x)}{q_i(x)^2}\right],
\end{aligned}$$

where $\pi_{ij} \stackrel{\text{def}}{=} \mathbb{E}_{p_j}\left[\frac{p_i(x)f(x)}{q_i(x)}\right]$. □

A.1.2 Estimating recall

In Section 3.4, we used the fact that the recall of system i, r_i, can be expressed as the recall of i within the pool, ν_i and the recall of the pool itself θ: $r_i = \theta \nu_i$:

$$\nu_i = \mathbb{E}_{x \sim \mathcal{Y}|Y}[g_i(x)] \qquad\qquad \theta = \mathbb{E}_{x \sim \mathcal{Y}}[g(x)],$$

where x is sampled under the distribution $p'(x|x \in Y)$ and $p'(x)$ respectively and $g(x) \stackrel{\text{def}}{=} \mathbb{I}[x \in \bigcup_{i=1}^{m} X_i] = \max_{j \in [1,m]} g_j(x)$ is the indicator function for x belonging to the pool.

Ideally, to estimate the pooled recall, ν_i, we need to take expectations with respect to $x \sim Y$. However, we only have samples drawn from individual X_i. To correct for this bias, we'll use a self-normalizing estimator for ν_i:

$$\hat{\nu}_i \stackrel{\text{def}}{=} \frac{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x) g_i(x)}{q(x)}}{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x)}{q(x)}},$$

where $p'(x) \propto p_0(x)$, $q(x) = \sum_{j=1}^{m} w_j p_j(x)$ and $w_j \geq 0$ are mixture parameters such that $\sum_{j=1}^{m} w_j = 1$.

The pool recall θ can be estimated as follows:

$$\hat{\theta} \stackrel{\text{def}}{=} \sum_{x \in \hat{Y}_0} g(x),$$

where $g(x) \stackrel{\text{def}}{=} \mathbb{I}\left[x \in \bigcup_{i=1}^{m} X_i\right] = \max_{j \in [1,m]} g_j(x)$.

Finally, we proposed the following estimator for recall r_i:

$$\hat{r}_i \stackrel{\text{def}}{=} \hat{\theta} \hat{\nu}_i.$$

Let's start by showing that ν_i is unbiased.

Theorem 3 (Statistical properties of $\hat{\nu}_i$)**.** *$\hat{\nu}_i$ is a consistent estimator of ν_i.*

Proof. We have that $p'_Y(x) = \frac{w(x)}{Z_Y}$. While we do not know the value of Z_Y, we can divide both the numerator and denominator of $\hat{\nu}_i$ by this quantity:

$$\begin{aligned}\hat{\nu}_i &= \frac{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x) g_i(x)}{Z_Y q(x)}}{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p_0(x)}{Z_Y q(x)}} \\ &= \frac{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x) g_i(x)}{q(x)}}{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x)}{q(x)}}.\end{aligned}$$

As the number of samples $n_i \to \infty$,

$$\begin{aligned}\mathbb{E}_X[\hat{\nu}_i] &= \mathbb{E}_X\left[\frac{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x) g_i(x)}{q(x)}}{\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x)}{q(x)}}\right] \\ &= \frac{\mathbb{E}_X\left[\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x) g_i(x)}{q(x)}\right]}{\mathbb{E}_X\left[\sum_{j=1}^{m} \frac{w_j}{n_j} \sum_{x \in \hat{Y}_j} \frac{p'_Y(x)}{q(x)}\right]}.\end{aligned}$$

Following similar arguments as in the proof of Theorem 2, the numerator and denominator are unbiased estimators of $\mathbb{E}_{x \sim \mathcal{Y}|Y}[g_i(x)]$ and $\mathbb{E}_{x \sim \mathcal{Y}|Y}[1] = 1$ respectively. Thus,

$$\begin{aligned}\mathbb{E}_X[\hat{\nu}_i] &= \mathbb{E}_{x \sim \mathcal{Y}|Y}[g_i(x)] \\ &= \nu_i.\end{aligned}$$

$\hat{\nu}_i$ is an unbiased estimator of ν_i.

□

Finally, we turn to studying $\hat{r}$:

Theorem 4 (Statistical properties of $\hat{r}_i$)**.** *$\hat{r}_i$ is an unbiased estimator of r_i with variance*

$$\operatorname{Var} \hat{r}_i = \theta \operatorname{Var} \hat{\nu}_i + \nu_i \operatorname{Var} \hat{\theta} + \operatorname{Var} \hat{\theta} \operatorname{Var} \hat{\nu}_i.$$

Proof. First, let's show that $r_i = \theta \nu_i$:

$$\begin{aligned} r_i &\stackrel{\text{def}}{=} \mathbb{E}_{x\sim\mathcal{Y}}[g_i(x)] \\ &= p'(Y_i) \\ &= p'(Y \wedge Y_i) \\ &= p'(Y)p'(Y_i|Y) \\ &= \mathbb{E}_{x\sim\mathcal{Y}}[g(x)]\mathbb{E}_{x\sim\mathcal{Y}|Y}[g_i(x)] \\ &= \theta\nu_i. \end{aligned}$$

From Theorem 3, we have that $\hat{\nu}_i$ is an unbiased estimator of ν_i. It is evident that $\hat{\theta}$ is an unbiased estimator of θ. $\hat{\nu}_i$ and $\hat{\theta}$ are estimated using independent samples ($\hat{Y}$ and $\hat{Y}_0$ respectively), and hence

$$\begin{aligned} \mathbb{E}_{Y_0,Y}[\hat{r}] &= \mathbb{E}_{Y_0,Y}[\hat{\theta}\hat{\nu}_i] \\ &= \mathbb{E}_{Y_0}[\hat{\theta}]\mathbb{E}_{Y}[\hat{\nu}_i] \\ &= \theta\nu_i \\ &= \hat{r}. \end{aligned}$$

By Lemma 1,

$$\operatorname{Var}\hat{r}_i = \theta \operatorname{Var}\hat{\nu}_i + \nu_i \operatorname{Var}\hat{\theta} + \operatorname{Var}\hat{\theta}\operatorname{Var}\hat{\nu}_i.$$

□

A.1.3 Picking heuristic w_{ij}.

A.1.4 Picking optimal number of samples for a new system

In Section 3.4.3, we outlined a method to pick the optimal number of samples to draw and evaluate for a new system: we pick the minimum number of samples n_m required to evaluate system m within a target variance using a conservative estimate of the variance of $\hat{\pi}_m^{\text{(joint)}}$. In particular, we use the following estimate for variance using the result from Theorem 2:

$$\widehat{\mathrm{Var}}\hat{\pi}_m = \sum_{j=1}^{m-1} \frac{w_j^2}{n_j} \sum_{x \in \hat{X}_j} \frac{1}{n_j} \left[\frac{p_i(x)^2 f(x)^2 - \pi_{ij} p_i(x) f(x) q_i(x)}{q_i(x)^2} \right] + \frac{w_m^2}{n_m} \sum_{x \in X_m} p_m(x) \left[\frac{p_m(x)}{q(x)} \right]^2,$$

where the first $m-1$ terms are an empirical estimate of variance and the last term is an upper bound on the variance. We note that the actual output of each system, X_j, and the samples drawn from previous systems, $\hat{X}_j$, is known. Thus, the only variable in computing $\widehat{\mathrm{Var}}\hat{\pi}_m$ is n_m. Furthermore, $\widehat{\mathrm{Var}}\hat{\pi}_m$ is a monotonically decreasing in n_m, so we can easily solve for the minimum number of samples required to estimate $\hat{\pi}_m^{\text{(joint)}}$ within a confidence interval ϵ by using the bisection method (Burden and Faires, 1985).

A.2 Basic probability lemmas

Lemma 1 (Mean and variance of the product of two random variables)**.** *Let x and y be two independent random variables with means μ_x and μ_y, and variances σ_x^2 and σ_y^2. Then, the estimator $z = xy$ has mean $\mu_x \mu_y$ and variance*

$$\sigma_z^2 = \sigma_x^2 \sigma_y^2 + \mu_x^2 \sigma_y^2 + \sigma_x^2 \mu_y^2.$$

Proof. If x and y are independent, $\mathbb{E}[xy] = \mathbb{E}[x]\mathbb{E}[y]$. Thus $\mathbb{E}[z] = \mu_x \mu_y$.

The variance of z can be calculated as follows:

$$\begin{aligned}
\mathrm{Var}(z) &= \mathbb{E}[z^2] - \mathbb{E}[z]^2 \\
&= \mathbb{E}[(xy)^2] - \mathbb{E}[xy]^2 \\
&= \mathbb{E}[x^2]\mathbb{E}[y^2] - \mathbb{E}[x]^2\mathbb{E}[y]^2 \\
&= (\sigma_x^2 + \mu_x^2)(\sigma_y^2 + \mu_y^2) - \mu_x^2 \mu_y^2 \\
&= \sigma_x^2 \sigma_y^2 + \mu_x^2 \sigma_y^2 + \sigma_x^2 \mu_y^2 + \mu_x^2 \mu_y^2 - \mu_x^2 \mu_y^2 \\
&= \sigma_x^2 \sigma_y^2 + \mu_x^2 \sigma_y^2 + \sigma_x^2 \mu_y^2.
\end{aligned}$$

□

Lemma 2 (Mean and variance of the ratio of two random variables)**.** *Let x and y be two random variables such that y is strictly positive (i.e. $y > 0$) with means μ_x and μ_y, variances σ_x^2 and σ_y^2. Then, the first-order Taylor approximation of $z = x/y$ has mean μ_x/μ_y. Furthermore, if x and y are the mean of a n_x and n_y independent random variables, the approximation error of using the first-order approximation goes to 0 as $n_x, n_y \to \infty$.*

Proof. This is a standard result in statistics. For completeness, we provide a proof below.

Let $f(x, y) = \frac{x}{y}$. Even if x and y are independent, $\mathbb{E}[f(x, y)]$ is not necessarily equal to $f(\mathbb{E}[x], \mathbb{E}[y])$. However, taking a first-order Taylor expansion around (μ_x, μ_y), we get

$$\begin{aligned}\mathbb{E}[f(x,y)] &\approx f(\mu_x, \mu_y) + f'_x(\mu_x, \mu_y)\mathbb{E}[x - \mu_x] + f'_y(\mu_x, \mu_y)\mathbb{E}[y - \mu_y] \\ &= \frac{\mu_x}{\mu_y}.\end{aligned}$$

We note that if x and y are the sum of independent random variables, then by the central limit theorem all moments of x and y greater than 1 go to 0 as $n_x, n_y \to \infty$.

□

Appendix B

Supplementary material for Chapter 4

B.1 Proofs

In this section, we provide proofs for the theorems stated in Chapter 4.

B.1.1 Main Theorem

In this section, we prove the main theorem (Theorem 1) in the Chapter 4 about the minimax optimal variance for an unbiased estimator. Theorem 1 will follow from the two following lemmas (Lemmas 3 and 4). First, we show in Lemma 3 that for all distributions with fixed σ_f^2, σ_a^2 and ρ, the variance of $\hat{\mu}_{\text{cv}}$ is constant and equal to: $\frac{1}{n}(\sigma_f^2(1-\rho^2)+\sigma_a^2)$. Then we give an explicit distribution, a Gaussian distribution, where *any* estimator yields at least this variance using the theory of sufficient statistics. Together, these show that the max variance of any estimator is at least the max variance of $\hat{\mu}_{\text{cv}}$.

As a reminder, the estimator is

$$\hat{\mu}_{\text{cv}} = \frac{1}{n}\sum_i y^{(i)} - \alpha g(z^{(i)}) \tag{B.1}$$

where $\alpha = \text{Cov}(f(z), g(z))$.

Lemma 3. *The variance of $\hat{\mu}_{cv}$ is always*

$$\frac{1}{n}(\sigma_f^2(1-\rho^2)+\sigma_a^2) \tag{B.2}$$

Proof. By the law of total variance, with respect to the draws of $z^{(i)}$,

$$\mathrm{Var}(\hat{\mu}_{\mathrm{cv}}) = \mathbb{E}_{z^{(i)}}[\mathrm{Var}(\hat{\mu}_{\mathrm{cv}}|z^{(i)})] + \mathrm{Var}_{z^{(i)}}(\mathbb{E}[\hat{\mu}_{\mathrm{cv}}|z^{(i)}]) \tag{B.3}$$

We will evaluate each of the two terms on the right hand side.

For the first term,

$$\mathbb{E}_{z^{(i)}}[\mathrm{Var}(\hat{\mu}_{\mathrm{cv}}|z^{(i)})] = \mathbb{E}_{z^{(i)}}\left[\mathrm{Var}\left(\frac{1}{n}\sum_i y^{(i)}|z^{(i)}\right)\right] \tag{B.4}$$

Because the human responses $Y(z^{(i)})$ are uncorrelated,

$$\begin{aligned}
\mathbb{E}_{z^{(i)}}[\mathrm{Var}(\hat{\mu}_{\mathrm{cv}}|z^{(i)})] &= \mathbb{E}_{z^{(i)}}\left[\frac{1}{n^2}\sum_i \mathrm{Var}(Y(z^{(i)}))|z^{(i)}\right] && \text{(B.5)}\\
&= \frac{1}{n}\mathbb{E}_z[\mathrm{Var}(Y(z))] && \text{(B.6)}\\
&= \frac{1}{n}\sigma_a^2 && \text{(B.7)}
\end{aligned}$$

For the second term,

$$\mathrm{Var}_{z^{(i)}}(\mathbb{E}[\hat{\mu}_{\mathrm{cv}}|z^{(i)}]) = \mathrm{Var}_{z^{(i)}}\left(\frac{1}{n}\sum_i f(z^{(i)}) - \alpha g(z^{(i)})\right) \tag{B.8}$$

Because the $z^{(i)}$ are sampled independently,

$$\mathrm{Var}_{z^{(i)}}(\mathbb{E}[\hat{\mu}_{\mathrm{cv}}|z^{(i)}]) = \frac{1}{n}\mathrm{Var}(f(z) - \alpha g(z)) \tag{B.9}$$
$$= \frac{1}{n}[\mathrm{Var}(f(z)) - 2\alpha\,\mathrm{Cov}(f(z), g(z)) + \alpha^2\,\mathrm{Var}(g(z))] \tag{B.10}$$

Note that $\mathrm{Var}(f(z)) = \sigma_f^2$, $\mathrm{Cov}(f(z), g(z)) = \alpha$, and $\mathrm{Var}(g(z)) = 1$ (since it is normalized). Thus,

$$\mathrm{Var}_{z^{(i)}}(\mathbb{E}[\hat{\mu}_{\mathrm{cv}}|z^{(i)}]) = \frac{1}{n}[\sigma_f^2 - 2\alpha^2 + \alpha^2] \tag{B.11}$$
$$= \frac{1}{n}[\sigma_f^2 - \alpha^2] \tag{B.12}$$

Since the correlation $\rho = \frac{\alpha}{\sigma_f \sigma_g} = \frac{\alpha}{\sigma_f}$,

$$\mathrm{Var}_{z^{(i)}}(\mathbb{E}[\hat{\mu}_{\mathrm{cv}}|z^{(i)}]) = \frac{1}{n}[\sigma_f^2 - \sigma_f^2\rho^2] \tag{B.13}$$
$$= \frac{1}{n}\sigma_f^2(1 - \rho^2) \tag{B.14}$$

Putting these two terms together, we find that,

$$\mathrm{Var}(\hat{\mu}_{\mathrm{cv}}) = \frac{1}{n}\sigma_a^2 + \frac{1}{n}\sigma_f^2(1 - \rho^2) \tag{B.15}$$
$$= \frac{1}{n}(\sigma_f^2(1 - \rho^2) + \sigma_a^2) \tag{B.16}$$

□

For the next lemma, we show that the worst-case variance for any estimator is at least that of $\hat{\mu}_{\mathrm{cv}}$. For this, we will define a simple Gaussian distribution and use the theory of sufficient statistics. We explicitly define a distribution over $f(z)$, $g(z)$, and $Y(Z) - f(z)$. In particular, we assume these are all Gaussian distributions with respective means, $\mu, 0, 0$, and variances, $\sigma_f^2, 1, \sigma_a^2$. Additionally, we assume that $f(z)$ and $g(z)$ have covariance α but

$Y(z) - f(z)$ is independent.

Lemma 4. *$\hat{\mu}_{cv}$ is the minimal variance unbiased estimate (MVUE) for the Gaussian distribution above.*

Proof. The proof is straightforward: we first show that $\hat{\mu}_{\text{cv}}$ is a sufficient statistic using the Fisher-Neyman factorization theorem, and then we apply the Lehman-Scheffe theorem.

For ease of notation, define $g_i = g(z^{(i)})$ and $y_i = y^{(i)}$. For the purposes of statistics, only μ is a parameter; the other "parameters" are known constants. Note that the pdf of the observed variables g_i and y_i is,

$$\prod_i c_1 \exp(-\frac{1}{2} \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}^T \begin{bmatrix} \sigma_f^2 + \sigma_a^2 & \alpha \\ \alpha & 1 \end{bmatrix}^{-1} \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}) \tag{B.17}$$

$$= c_2 \exp(-\frac{1}{2} \sum_i \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}^T \begin{bmatrix} \sigma_f^2 + \sigma_a^2 & \alpha \\ \alpha & 1 \end{bmatrix}^{-1} \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}) \tag{B.18}$$

Thus, with the Fisher-Neyman factorization theorem, it suffices to show that the exponetiated term T decomposes as a sum of a function that only depends on the data and a function that only depends on $\hat{\mu}_{\text{cv}}$ and μ.

$$T = \sum_i \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}^T \begin{bmatrix} \sigma_f^2 + \sigma_a^2 & \alpha \\ \alpha & 1 \end{bmatrix}^{-1} \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix} \tag{B.19}$$

Letting c_3 be the inverse determinant (which is constant),

$$T = c_3 \sum_i \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix}^T \begin{bmatrix} 1 & -\alpha \\ -\alpha & \sigma_f^2 + \sigma_a^2 \end{bmatrix} \begin{bmatrix} (y_i - \mu) \\ g_i \end{bmatrix} \tag{B.20}$$

$$= c_3 \left[\sum_i (y_i - \mu)^2 - 2\alpha \sum_i (y_i - \mu) g_i + (\sigma_f^2 + \sigma_a^2) \sum_i g_i^2 \right] \tag{B.21}$$

$$= c_3 \left[\sum_i y_i^2 - 2\mu \sum_i y_i + n\mu^2 - 2\alpha \sum_i y_i g_i + 2\alpha\mu \sum_i g_i + (\sigma_f^2 + \sigma_a^2) \sum_i g_i^2 \right] \tag{B.22}$$

$$= -2c_3\mu \left[\sum_i y_i - \alpha \sum_i g_i \right] + c_3 n\mu^2 + c_3 \left[\sum_i y_i^2 - 2\alpha \sum_i y_i g_i + (\sigma_f^2 + \sigma_a^2) \sum_i g_i^2 \right] \tag{B.23}$$

$$= -2nc_3\mu\hat{\mu}_{\text{cv}} + c_3 n\mu^2 + c_3 \left[\sum_i y_i^2 - 2\alpha \sum_i y_i g_i + (\sigma_f^2 + \sigma_a^2) \sum_i g_i^2 \right] \tag{B.24}$$

Thus, we see the decomposition into the function of only the data on the right and only μ and $\hat{\mu}_{\text{cv}}$ on the left. Thus, $\hat{\mu}_{\text{cv}}$ is a sufficient statistic.

Further, $\hat{\mu}_{\text{cv}}$ is an unbiased estimate of μ since $\mathbb{E}[g_i] = 0$ and $\mathbb{E}[y_i] = \mu$.

Further, since $\hat{\mu}_{\text{cv}}$ is normally distributed with mean dependent on μ, it is complete.

Thus, by the Lehmann-Scheffe theorem, $\hat{\mu}_{\text{cv}}$ is the minimal variance unbiased estimate (MVUE).

□

Theorem 1. *Among all unbiased estimators that are functions of $y^{(i)}$ and $g(z^{(i)})$, and for all distributions with a given σ_f^2, σ_a^2, and α,*

$$\text{Var}(\hat{\mu}_{cv}) = \frac{1}{n}(\sigma_f^2(1 - \rho^2) + \sigma_a^2), \tag{B.25}$$

and no other estimator has a lower worst-case variance.

Proof. From Lemma 3 we have that the max variance of $\hat{\mu}_{\text{cv}}$ over all distributions with fixed variances, is exactly,

$$\frac{1}{n}(\sigma_f^2(1-\rho^2)+\sigma_a^2) \tag{B.26}$$

Further, from Lemma 4, we know that $\hat{\mu}_{\text{cv}}$ is the MVUE for a particular class of distributions, thus, any estimator has a larger max variance over all distributions.

Combining these two facts, we get that the minimax variance is the variance of $\hat{\mu}_{\text{cv}}$. □

B.1.2 Added Bias

Proposition 1. *The estimator in Algorithm 1 has $O(1/n)$ bias.*

Proof. The bias B is

$$B = |\mathbb{E}[\widetilde{\mu}] - \mu| \tag{B.27}$$

$$= \left|\mathbb{E}[\frac{1}{n}\sum_i y^{(i)} - \hat{\alpha} g(z^{(i)})] - \mu\right| \tag{B.28}$$

Since $\mathbb{E}[y^{(i)}] = \mu$,

$$B = \left|\mu - \frac{1}{n}\sum_i \mathbb{E}[\hat{\alpha} g(z^{(i)})] - \mu\right| \tag{B.29}$$

$$= \left|\frac{1}{n}\sum_i \mathbb{E}[\hat{\alpha} g(z^{(i)})]\right| \tag{B.30}$$

$$= \left|\frac{1}{n^2}\sum_{i,j} \mathbb{E}[(y^{(j)} - \overline{y}) g(z^{(j)}) g(z^{(i)})]\right| \tag{B.31}$$

$$= \left|\frac{1}{n^2}\sum_{i,j} \mathbb{E}[y^{(j)} g(z^{(j)}) g(z^{(i)})] - \frac{1}{n^3}\sum_{i,j,k} \mathbb{E}[y^{(k)} g(z^{(j)}) g(z^{(i)})]\right| \tag{B.32}$$

Because $Y(z)$ is independent and has mean $f(z)$,

$$B = \left| \frac{1}{n^2} \sum_{i,j} \mathbb{E}[f(z^{(j)})g(z^{(j)})g(z^{(i)})] - \frac{1}{n^3} \sum_{i,j,k} \mathbb{E}[f(z^{(k)})g(z^{(j)})g(z^{(i)})] \right| \tag{B.33}$$

Because $g(z)$ is mean zero and the $z^{(i)}$ are drawn independently,

$$\begin{aligned} B &= \left| \frac{1}{n^2} \sum_{i} \mathbb{E}[f(z^{(i)})g(z^{(i)})^2] - \frac{1}{n^3} \sum_{i,k} \mathbb{E}[f(z^{(k)})g(z^{(i)})^2] \right| && \text{(B.34)} \\ &= \left| \frac{1}{n^2} \sum_{i} O(1) - \frac{1}{n^3} \sum_{i,k} O(1) \right| && \text{(B.35)} \\ &= \left| \frac{1}{n^2} O(n) - \frac{1}{n^3} O(n^2) \right| && \text{(B.36)} \\ &= \left| O\left(\frac{1}{n}\right) - O\left(\frac{1}{n}\right) \right| && \text{(B.37)} \\ &= O\left(\frac{1}{n}\right) && \text{(B.38)} \end{aligned}$$

$\square$

Bibliography

H. Adel, B. Roth, and H. Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

L. von Ahn and L. A. Dabbish. 2004. Labeling images with a computer game. In *Conference on Human Factors in Computing Systems (CHI)*.

G. Angeli, J. Tibshirani, J. Y. Wu, and C. D. Manning. 2014. Combining distant and partial supervision for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. A. Aslam, V. Pavlu, and E. Yilmaz. 2006. A statistical method for system evaluation using incomplete judgments. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 541–548.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *User Interface Software and Technology*, pages 33–42.

M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. 2010. Soylent: a word processor with a crowd inside. In *Symposium on User Interface Software and Technology*, pages 313–322.

R. Brandow, K. Mitze, and L. F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31:675–685.

C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. 2007. Bias and the limits of pooling for large collections. In *ACM Special Interest Group on Information Retreival (SIGIR)*.

C. Buckley and E. M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 25–32.

R. L. Burden and J. D. Faires. 1985. *Numerical Analysis (3rd ed.)*. PWS Publishers.

N. Carlini and D. Wagner. 2016. Defensive distillation is not robust to adversarial examples. *arXiv*.

N. Carlini and D. Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv*.

G. Casella and R. L. Berger. 1990. *Statistical Inference*. Wadsworth and Brooks.

N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

A. Chaganty, A. Paranjape, P. Liang, and C. Manning. 2017a. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. T. Chaganty, A. Paranjape, J. Bolton, M. Lamm, J. Lei, A. See, K. Clark, Y. Zhang, P. Qi, and C. D. Manning. 2017b. Stanford at TAC KBP 2017: Building a trilingual relational knowledge graph. In *Text Analytics Conference*.

X. Chai, L. Deng, Q. Yang, and C. X. Ling. 2004. Test-cost sensitive naive Bayes classification. In *International Conference on Data Mining*, pages 51–58.

C. Chang, R. Yang, L. Chen, X. Zhou, and K. Yu. 2017. Affordable on-line dialogue policy learning. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 223–231.

J. Cheng and M. S. Bernstein. 2015. Flock: Hybrid Crowd-Machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 600–611.

W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng. 2011. Unbiased online active learning in data streams. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 195–203.

C. W. Cleverdon. 1962. Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. In *ASLIB*.

C. W. Cleverdon. 1967. The cranfield tests on index language devices. In *ASLIB*.

A. Cohan and N. Goharian. 2016. Revisiting summarization evaluation for scientific articles. In *Language Resources and Evaluation Conference (LREC)*.

J. M. Conroy and H. T. Dang. 2008. Mind the gap : Dangers of divorcing evaluations of summary content from linguistic quality. In *International Conference on Computational Linguistics (COLING)*, pages 145–152.

G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. 1998. Efficient construction of large test collections. In *ACM Special Interest Group on Information Retreival (SIGIR)*.

R. Coulom. 2007. Computing elo ratings of move patterns in the game of go. *Computer Games Workshop*.

C. Culy and S. Z. Riehemann. 2003. The limits of n-gram translation evaluation metrics. In *MT Summit IX*, pages 71–78.

P. Dai, Mausam, and D. S. Weld. 2010. Decision-theoretic control of crowd-sourced workflows. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

H. T. Dang. 2006. Overview of DUC 2006. In *Document Understanding Conference*.

H. T. Dang. 2016. Cold start knowledge base population at TAC KBP 2016. *Text Analytics Conference*.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

M. Denkowski and A. Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Workshop on Statistical Machine Translation*.

P. Donmez and J. G. Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Conference on Information and Knowledge Management (CIKM)*, pages 619–628.

O. Dusek, J. Novikova, and V. Rieser. 2017. Referenceless quality estimation for natural language generation. *arXiv*.

J. Ellis, J. Getman, H. Simpson, K. Griffitt, H. T. Dang, R. Grishman, H. Ji, C. DePrince, T. Riese, and N. Kuster. 2015. TAC KBP 2015 slot descriptions. *Linguistic Data Consortium*.

J. Ellis, X. Li, K. Griffitt, and S. M. Strassel. 2012. Linguistic resources for 2012 knowledge base population evaluations. *Text Analytics Conference*.

S. Esmeir and S. Markovitch. 2007. Anytime induction of cost-sensitive trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 425–432.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1156–1165.

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Association for Computational Linguistics (ACL)*, pages 363–370.

J. Foster, O. Cetinoglu, J. Wagner, J. L. Roux, J. Nivre, D. Hogan, and J. VanGenabith. 2011. From news to comment: Resources and benchmarks for parsing the language of Web 2.0. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

T. Gao and D. Koller. 2011. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1062–1070.

D. Golovin, A. Krause, and D. Ray. 2010. Near-optimal Bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 766–774.

I. J. Goodfellow, J. Shlens, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

E. Greensmith, P. L. Bartlett, and J. Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 5:1471–1530.

R. Greiner, A. J. Grove, and D. Roth. 2002. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174.

S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

H. V. Halteren and S. Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 57–64.

S. Han, J. Bang, S. Ryu, and G. G. Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 129–133.

D. K. Harman. 1992. Overview of the first TREC text retrieval conference. In *Text Retrieval Conference*.

D. K. Harman. 1993. The first text retrieval conference (trec-1) rockville, md, u.s.a., 4-6 november, 1992. *Information Processing and Management*, 29:411–414.

L. He, M. Lewis, and L. Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D. Helmbold and S. Panizza. 1997. Some label efficient learning results. In *Conference on Learning Theory (COLT)*, pages 218–230.

K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

H. Ji, R. Grishman, and H. Trang Dang. 2011. Overview of the TAC 2011 knowledge base population track. In *Text Analytics Conference*.

R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

K. S. Jones and C. V. Rijsbergen. 1975. Report on the need for and provision of an “ideal test collection. *Information Retrieval Test Collection*.

A. Kalyanpur, B. K. Boguraev, S. Patwardhan, J. W. Murdock, A. Lally, C. A. Welty, J. M. Prager, B. Coppola, A. Fokoue-Nkoutche, L. Zhang, Y. Pan, and Z. M. Qui. 2012. Structured data and inference in deepqa. *IBM Journal of Research and Development*, 56:351–364.

T. Kočisky, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. 2017. The NarrativeQA reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

L. Kocsis and C. Szepesvári. 2006. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, pages 282–293.

N. Kokkalis, T. Köhn, C. Pfeiffer, D. Chornyi, M. S. Bernstein, and S. R. Klemmer. 2013. Emailvalet: Managing email overload through private, accountable crowdsourcing. In *Conference on Computer Supported Cooperative Work*, pages 1291–1300.

R. Krishna, K. Hata, S. Chen, J. Kravitz, D. A. Shamma, L. Fei-Fei, and M. S. Bernstein. 2016. Embracing error to enable rapid crowdsourcing. In *Conference on Human Factors in Computing Systems (CHI)*.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105.

N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. 2009. Attribute and simile classifiers for face verification. In *International Conference on Computer Vision (ICCV)*, pages 365–372.

W. S. Lasecki, Y. C. Song, H. Kautz, and J. P. Bigham. 2013. Real-time crowd labeling for deployable activity recognition. In *Conference on Computer Supported Cooperative Work*, pages 1203–1212.

A. Lavie and M. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23.

C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B. Lee. 2012. Twiner: named entity recognition in targeted twitter stream. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 721–730.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML)*.

C. Lin and M. Rey. 2004. Looking for a few good metrics: ROUGE and its evaluation. In *NTCIR Workshop*.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll'ar, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755.

A. Liu, S. Soderland, J. Bragg, C. H. Lin, X. Ling, and D. S. Weld. 2016a. Effective crowd annotation for relation extraction. In *North American Association for Computational Linguistics (NAACL)*, pages 897–906.

C. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016b. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. 2017a. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Association for Computational Linguistics (ACL)*.

R. T. Lowe, N. Pow, I. Serban, L. Charlin, C. Liu, and J. Pineau. 2017b. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue and Discourse*, 8.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. 2011. Learning word vectors for sentiment analysis. In *Association for Computational Linguistics (ACL)*.

I. Mani, G. Klein, L. Hirschman, T. Firmin, D. House, and B. Sundheim. 1999. The TIPSTER SUMMAC text summarization evaluation. In *European Association for Computational Linguistics (EACL)*.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.

J. Mayfield and T. Finin. 2012. Evaluating the quality of a knowledge base populated from text. In *Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

D. McClosky. 2010. *Any domain parsing: automatic domain adaptation for natural language parsing*. Ph.D. thesis, Brown University.

K. Mckeown, R. J. Passonneau, D. K. Elson, and J. Hirschberg. 2005. Do summaries help? a task-based evaluation of multi-document summarization. In *ACM Special Interest Group on Information Retreival (SIGIR)*.

G. A. Miller and J. G. Beebe-Center. 1956. Some psychological methods for evaluating the quality of translations. *Mechanical Translation*, 3:73–80.

V. Mnih, C. Szepesv'ari, and J. Audibert. 2008. Empirical berstein stopping. In *International Conference on Machine Learning (ICML)*.

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

J. Novikova, O. Dušek, A. C. Curry, and V. Rieser. 2017. Why we need new evaluation metrics for NLG. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. B. Owen. 2013. *Monte Carlo theory, methods and examples*.

M. Pagliardini, P. Gupta, and M. Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv*.

J. Paisley, D. M. Blei, and M. I. Jordan. 2012. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning (ICML)*, pages 1363–1370.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*.

R. J. Passonneau and B. Carpenter. 2014. The benefits of a model of annotation. In *Association for Computational Linguistics (ACL)*.

R. J. Passonneau, A. Nenkova, K. McKeown, and S. Sigelman. 2005. Applying the pyramid method in DUC 2005. In *Document Understanding Conference*.

R. Paulus, C. Xiong, and R. Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

E. Pavlick, H. Ji, X. Pan, and C. Callison-Burch. 2016. The gun violence database: A new task and data set for NLP. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1018–1024.

J. R. Pierce. 1970. Whither speech recognition? *Journal of the Acoustical Society of America*, 47:1616–1617.

B. Plank. 2011. *Domain adaptation for parsing*. Ph.D. thesis, University of Groningen.

B. Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. *arXiv*.

M. Poesio, J. Chamberlain, U. Kruschwitz, L. Robaldo, and L. Ducceschi. 2013. Phrase Detectives: Utilizing collective intelligence for internet-scale language resource creation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

R. Ranganath, S. Gerrish, and D. Blei. 2014. Black box variational inference. In *Artificial Intelligence and Statistics (AISTATS)*, pages 814–822.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Association for Computational Linguistics (ACL)*.

B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. 2018. Do CIFAR-10 classifiers generalize to CIFAR-10? *arXiv*.

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 2(10):377–392.

B. D. Ripley. 2009. *Stochastic simulation*. John Wiley & Sons.

T. Sakai and N. Kando. 2008. On information retrieval metrics designed for evaluation with incomplete relevance assessments. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 447–470.

G. Salton and M. E. Lesk. 1965. The SMART automatic document retrieval systems—an illustration. *Communications of the ACM*, 8(6):391–398.

D. Sculley. 2007. Online active learning methods for fast label-efficient spam filtering. In *Conference on Email and Anti-spam (CEAS)*.

A. See, P. J. Liu, and C. D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics (ACL)*.

B. Settles. 2010. Active learning literature survey. Technical report, University of Wisconsin, Madison.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation in the Americas*, pages 223–231.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Tan, F. Wei, N. Yang, W. Lv, and M. Zhou. 2018. S-Net: From answer extraction to answer generation for machine reading comprehension. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

K. Toutanova and C. Brockett. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 340–350.

D. Vannella, D. Jurgens, D. Scarfini, D. Toscani, and R. Navigli. 2014. Validating and extending semantic knowledge bases using video games with a purpose. In *Association for Computational Linguistics (ACL)*, pages 1294–1304.

R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

E. M. Voorhees. 2007. Trec: Continuing information retrieval's tradition of experimentation. *Communications of the ACM*, 50(11):51–54.

W. E. Webber. 2010. *Measurement in Information Retrieval Evaluation*. Ph.D. thesis, University of Melbourne.

K. Weiss, T. M. Khoshgoftaar, and D. Wang. 2016. A survey of transfer learning. *Journal of Big Data*, 3.

D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Computational Natural Language Learning (CoNLL)*.

J. White, T. O'Connell, and F. O'Mara. 1994. The ARPA MT evaluation methodologies: evaluation, lessons, and future approaches. In *First Conference of the Association for Machine Translation in the Americas*.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

E. Yilmaz, E. Kanoulas, and J. A. Aslam. 2008. A simple and efficient sampling method for estimating AP and NDCG. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 603–610.

J. Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *ACM Special Interest Group on Information Retreival (SIGIR)*.