How Good Are My Predictions? Efficiently Approximating Precision-Recall Curves for Massive Datasets

Ashish Sabharwal and **Hanie Sedghi**
Allen Institute for Artificial Intelligence (AI2), Seattle, WA, U.S.A.
AshishS,HanieS@allenai.org

Abstract

Large scale machine learning produces massive datasets whose items are often associated with a confidence level and can thus be ranked. However, computing the precision of these resources requires human annotation, which is often prohibitively expensive and is therefore skipped. We consider the problem of cost-effectively approximating precision-recall (PR) or ROC curves for such systems. Our novel approach, called PAULA, provides theoretically guaranteed lower and upper bounds on the underlying precision function while relying on only $\mathrm{O}(\log N)$ annotations for a resource with N items. This contrasts favorably with $\Theta(\sqrt{N \log N})$ annotations needed by commonly used sampling based methods. Our key insight is to capitalize on a natural monotonicity property of the underlying confidence-based ranking. PAULA provides tight bounds for PR curves using, e.g., only 17K annotations for resources with 200K items and 48K annotations for resources with 2B items. We use PAULA to evaluate a subset of the much utilized PPDB paraphrase database and a recent Science knowledge base.

1 INTRODUCTION

Precision-recall (PR) curves and receiver operating characteristic (ROC) curves play a fundamental role in understanding the behavior of a variety of systems in the presence of uncertainty. These curves are frequently used in machine learning (ML), electrical engineering, radiology, and many other fields to study the performance of a binary prediction system as a function of a control parameter. As the parameter is varied, one is often able to increase the precision (or decrease the false positive rate) of the system at the expense of lower recall (also known as the true positive rate or sensitivity). PR and ROC curves make this precision-recall tradeoff explicit, enabling users to adaptively choose a parameter value based on the needs of their application, or to compare two systems across the entire spectrum of the control parameter rather than at a single design point. From these curves, one can also "read off" summary performance metrics such as area under the curve (AOC) for classification algorithms, precision of the top k results (Prec@k) or average precision (AP) for ranking algorithms in applications such as Web search, and the F1 score for balancing precision and recall.

A key challenge in this space is cost-effectively approximating PR and ROC curves for massive resources produced by ML systems, when annotating whether an item in the resource is correct or not is expensive. For instance, consider the paraphrase database PPDB (Ganitkevitch et al., 2013; Pavlick et al., 2015) with 169M automatically computed pairs of English paraphrases. Each item in PPDB is associated with a confidence level, with the understanding that higher confidence correlates with being a valid paraphrase. However, the overall precision of PPDB, not to mention its PR tradeoff as a function of the confidence level, is unknown. Instead, PPDB is pre-packaged into six sizes (S, M, L, XL, etc.), ranging from the smallest package containing the highest ranking 6.8M pairs to the largest containing all 169M pairs. Similarly, the NELL system (Mitchell et al., 2012) has collected over 50M beliefs of which 3M are identified as more reliable. Again, its PR tradeoff is not available. Even smaller resources such as a semi-automatically generated Science KB with 217K triples (Dalvi et al., 2017) do not come with an associated PR curve. The bottleneck is the high cost of human annotation.

To address this bottleneck, we propose a novel method, called PAULA for *Precision Approximation Using Logarithmic Annotations*, for cost-effectively (in terms of an-

notations) computing a constant-factor approximation of the underlying *precision function* $p(r)$, which assigns a precision value (i.e., the fraction of items that are correct) to the top r items in a large, unannotated, ranked list $T = (t_1, \ldots, t_N)$. Note that any resource where each item is associated with a correctness confidence can be viewed as a ranked list. PAULA is able to obtain a pointwise $(1+\epsilon)$-approximation of p at a large number of points of interest (and roughly a pointwise $(1+\epsilon)^2$-approximation of entire p) using only roughly $\Delta \log_{1+\epsilon} N$ annotations in total (cf. Theorem 3), where Δ is a monotonicity parameter that is typically in the range 50 to 100. For the Science KB with 217K items, for instance, this translates into 17K annotations when $\epsilon = 0.03$. For a resource with 2B items, PAULA would need only 47K annotations.

The new insight is that one can leverage a natural property of p (and of derived metrics such as PR and ROC curves, to which our results also apply; see Sec. 2), that they are monotonic—when viewed through the lens of a sufficiently wide sliding window and looking beyond the initial few results in T where p is typically more erratic.

To put the benefit of this approach in perspective, we note that naïvely computing $p(r)$ exactly even at a single value of rank r is impractical, as it requires annotating all of the top r items. A common method to approximate PR curves is to randomly draw s samples $T' \subseteq_R T$, annotate T', create the PR curve for T', and use it as a surrogate for the actual PR curve for T. This has the advantage of providing a PR curve that becomes more and more accurate (due to more samples) as we go down the ranked list. We show (cf. Section 4.4.3), however, that it requires $s = \Theta(\sqrt{N \log N})$ annotations to achieve a constant factor approximation of p. E.g., for the 217K Science KB, it requires 47K annotations for a 95% confidence when $\epsilon = 0.03$ as opposed to only 17K annotations needed by PAULA, and this gap widens as N grows.

To analyze the behavior of PAULA, we first derive a general result (cf. Theorem 1) that under a monotonicity assumption, *it is sufficient to estimate p at only logarithmically many points* in T. We formalize the assumption and discuss its validity in Sections 4.1 and 4.2.

Building upon this, we consider a refined *stratified sampling* approach that requires only $\Theta(\log N \log\log N)$ annotations (Section 4.4.1, Theorem 2). However, while stratification helps reduce the number of annotations, the probabilistic approach results in cumulative errors, requiring one to be much more precise at estimating p at each individual point. We then develop our main contribution, the PAULA algorithm, in Section 4.4.2, leveraging a stronger form of monotonicity. Our estimates reveal that in typical use cases, *PAULA requires an order of magnitude fewer annotations than the random sampling baseline and half the annotations needed by our refined logarithmic stratified sampling approach.*

We evaluate PAULA on two large resources mentioned earlier: the PPDB dataset from natural language research and the Science KB for common facts about elementary level science. Our experiments show that the lower and upper bounds provided by PAULA are very close in practice—to each other and to the ground truth, when available. The method, being logarithmic and deterministic, easily scales to larger resources.

1.1 RELATED WORK

Much of the existing work in this area has focused on small ranked lists (e.g., the top 10 or 100 Web search results), on aggregating across multiple ranked lists (e.g., multiple search queries), and on summary statistics (e.g., Prec@k for some small k of interest, average precision AP, discounted cumulative gain DCG, etc.). In contrast, our challenge is to compute the entire precision function (and the associated PR, ROC, and precision-yield curves) for a single, large, ranked list. From the entire curve, it is then easy to "read off" various summary statistics.

There has been much interest in random sampling based methods to compute a particular summary statistic of interest (e.g., AP), where the focus is on choosing non-uniform samples in a way that minimizes variance. While very effective for that particular statistic, these methods require different sets of annotations to minimize variance for different metrics (e.g., Prec@k, Prec@k' for $k' \neq k$, Gain@k, DCG, etc.). Our method, on the other hand, relies on a deterministic set of annotations for the entire curve, and hence for any derived summary metrics.

Carterette et al. (2006) focus on the AP metric in the information retrieval (IR) setting, and propose a method to design a test subset that measures AP with high confidence. They choose documents based on the benefit they would provide when fully ranking the system.

Yilmaz et al. (2008) consider large scale retrieval evaluation using incomplete relevance judgments, focusing on a simple yet efficient method. Unlike prior work, they provide confidence intervals for the inferred AP and improve upon two proposals, one that is accurate but complex (Aslam et al., 2006) and another called InfAP that is simple but less efficient (Yilmaz and Aslam, 2006).

Kanoulas (2015) provides a short survey of methods, measures, and designs used in the field of IR to evaluate the quality of search algorithms against collections of observations, considering the use of relevance and observable user behavior to compute search quality.

Schnabel et al. (2016) provide an interesting, comprehensive perspective, recasting nearly all previous approaches with guarantees as Monte Carlo estimation of an expectation, focusing on variance reduction via importance sampling. Their domain of interest is Web search, where importance is placed on the top 10 or so ranked results, aggregation across multiple queries, and specific summary metrics that are used to guide the sampling strategy so as to minimizes variance.

2 PRELIMINARIES

Consider the ranked output $T = (t_1, t_2, \ldots, t_N)$ of an algorithm $\mathcal{A}$, where each t_i comes from some universe U (e.g., all documents on the Web, all paraphrase pairs, all subject-verb-object triples, etc.). Each item $u \in U$ is associated with an unknown true label $v(u) \in \{0, 1\}$ that captures the semantics of some underlying task (e.g., whether a document is relevant to a query, whether a pair of phrases is a linguistic paraphrase, whether a triple denotes a true fact, etc.). The **precision function of $\mathcal{A}$**, $p : [N] \to [0, 1]$, maps each rank $r \in [N]$ to the fraction of the top r items in T that are positive, i.e., labeled as 1:

$$p(r) = \frac{1}{r} \sum_{i=1}^{r} v(t_i) \tag{1}$$

where we omit $\mathcal{A}$ from the notation for brevity. The commonly used metric Prec@k for a specific k (e.g., Prec@10) is simply $p(k)$. The associated *recall of* $\mathcal{A}$ at rank r is the fraction of all positive items $U_+ \subseteq U$ that appear among $t_1, \ldots, t_r$; that is, $\frac{r\ p(r)}{|U_+|}$. Since U_+ is unknown in many ranking applications, it is common to instead use its unnormalized variant called *yield*, defined as $\sum_{i=1}^{r} v(t_i)$, which equals $r\ p(r)$.

A plot of $p(r)$ vs. recall (or yield) at r is commonly referred to as the *precision-recall* (or *precision-yield*, resp.) curve for $\mathcal{A}$. One can similarly define a variety of metrics that are derivable from $p(r), r, U_+$, and U, such as Gain@k, accuracy, F1, true positive rate (TPR), false positive rate (FPR), specificity, sensitivity, etc. The ROC curve refers to a plot of FPR vs. TPR, and average precision (AP) is the area under the PR curve. Fawcett (2006) and Majnik and Bosnic (2013) provide good surveys of such metrics, while Davis and Goadrich (2006) explore interesting relationships between PR and ROC curves.

Our goal in this work is to efficiently approximate p, which is a fundamental building block for all of the above metrics—indeed, a pointwise-approximation of p allows one to approximately compute all of these metrics.

Importantly, we operate under a setting where obtaining true labels $v(t_i)$ is costly (e.g., requires a system simulation, human annotation, crowdsourcing, etc.). We aim to *compute a pointwise-approximation of p with as few annotations of true labels $v(t_i)$* as possible.

2.1 POINT ESTIMATES: RANDOM SAMPLING

A simple way to obtain an unbiased point estimate of $p(r)$ for a *fixed* rank r is via random sampling: Sample (with repetition) z indices J independently and uniformly from $\{1, 2, \ldots, r\}$ and compute the empirical average $\tilde{p}(r) = \frac{1}{z} \sum_{j \in J} v(t_j)$. Then $\tilde{p}(r)$ is an unbiased estimator of $p(r)$ and one can apply tail inequalities such as the two-sided Hoeffding bound (Hoeffding, 1963) to compute how tight the estimate is:

$$\Pr[|\tilde{p}(r) - p(r)| \geq \epsilon p(r)] \leq 2 \exp\left(-2z\epsilon^2 p(r)^2\right)$$

For a $1 - \delta$ confidence in the outcome (e.g., a 95% confidence would mean $\delta = 0.05$), it suffices to have $2 \exp\left(-2z\epsilon^2 p(r)^2\right) \leq \delta$, which translates into needing:

$$z \geq \frac{1}{2\epsilon^2 p(r)^2} \ln \frac{2}{\delta} \tag{2}$$

samples in order to achieve a $(1 + \epsilon)$-approximation of $p(r)$. A complementary way of viewing the same result is that the use of z samples results in the following $1 - \delta$ confidence interval (e.g., a 95% confidence interval) which can be used for error bars around the estimate:

$$\tilde{p}(r) \pm \sqrt{\frac{1}{2z} \ln \frac{2}{\delta}} \tag{3}$$

3 PRECISION WITH $O(\sqrt{N \log N})$ ANNOTATIONS

A common practice to compute an approximation of the entire precision function p, not only its value at one point, is to draw s uniform random samples T' from the ranked list T, compute the precision function p' for T', and use this as a surrogate for p as $p(r) \approx p'(\frac{rs}{N})$. How close is $p'(\frac{rs}{N})$ an approximation of $p(r)$ is determined by the number $z = \frac{rs}{N}$ of the s random samples that are expected to land in the range $t_1, \ldots, t_r$. We provide a formal analysis of this approach below.

Eqn. (3) indicates that the approximation error starts off large (since we observe very few samples when r is small) and decreases as r increases, scaling proportionally to $1/\sqrt{r}$. To compare this method to our proposal for obtaining a pointwise $(1 + \alpha)$-approximation for entire p, we ask the following question: Given s and N, what is the smallest $n_{\alpha,s} \leq N$ such that for all $r \geq n_{\alpha,s}$, $p'(\frac{rs}{N})$ is a $(1 + \alpha)$-approximation of $p(r)$?

We apply Eqn. (2) to each of the N points, bounding the correctness confidence by $\delta' = \delta/N$ at each point and

taking the union bound to guarantee an overall correctness confidence of δ. This yields the requirement that $n_{\alpha,s}s/N$ must be at least $\frac{1}{2\alpha^2 p(r)^2}\ln\frac{2}{\delta'}$. The smallest $n_{\alpha,s}$ satisfying this is:

$$n_{\alpha,s} = \frac{N}{2s\alpha^2 p(r)^2}\ln\frac{2N}{\delta} \tag{4}$$

Thus, we obtain a $(1+\alpha)$-approximation after roughly the first $\frac{N\ln N}{s\alpha^2}$ points.

To obtain a $(1+\alpha)$-approximation of the *entire* precision function, even for low values of r, one possibility is to annotate all of $t_1,\ldots,t_{n_{\alpha,s}}$, in addition to s uniform random samples overall.[1] This would mean annotating $s+n_{\alpha,s}$ items in total. It can be verified that this expression is minimized when s is chosen such that $s = n_{\alpha,s}$, in which case the total number of annotations needed is:

$$\sqrt{\frac{2N}{\alpha^2 p(r)^2}\ln\frac{2N}{\delta}} \tag{5}$$

This expression grows asymptotically as $\Theta(\sqrt{N\log N})$. The methods developed in the next section achieve the same approximation with only $\mathrm{O}(\log N)$ annotations.

4 PRECISION WITH $\mathrm{O}(\log N)$ ANNOTATIONS

Exactly computing the entire precision function p, in particular computing $p(N)$, requires annotating N labels, which can be prohibitive for massive datasets. To alleviate this, we develop here a novel method called PAULA to efficiently obtain, for any $\epsilon > 0$, a pointwise $\gamma(1+\epsilon)$-approximation of p with roughly only $\Delta\log_{1+\epsilon}N$ deterministically chosen annotation points, where Δ is a constant capturing the level of monotonicity in the underlying data and γ is slightly larger than $1+\epsilon$.

4.1 LOCAL PRECISION AND MONOTONICITY

We start by defining a Δ-local variant of p and two related notions of monotonicity. Let $T = (t_1, t_2, \ldots, t_N)$ be the ranked output of an algorithm $\mathcal{A}$ with (unknown) true labels $v(t_i) \in \{0,1\}$ and precision function p.

Definition 1. *Let $\Delta \in \mathbb{N}^+$. The **Δ-precision of $\mathcal{A}$** is a function $p_\Delta : [N] \to [0,1]$ defined as:*

$$p_\Delta(r) = \begin{cases} \frac{1}{r}\sum_{i=1}^{r} v(t_i) & \text{if } r \le \Delta \\ \frac{1}{\Delta}\sum_{i=r-\Delta+1}^{r} v(t_i) & \text{otherwise} \end{cases} \tag{6}$$

where we omit $\mathcal{A}$ as before for brevity of notation.

[1] There is a small overlap when counting this way, which we ignore for simplicity of exposition.

Δ-precision may be viewed as a smoothed version of the true label sequence $v(t_1), v(t_2), \ldots, v(t_N)$. Although the actual label sequence, being a sequence of 0s and 1s, is bound to be erratic, we expect the density of 1s in this sequence to decrease as i increases. In other words, a sufficiently smoothed out variant of the true label sequence should be non-increasing, except perhaps for the initial few values. Formally, we use the following two characterizations of monotonicity:

Definition 2 (Weak Monotonicity)**.** *For $m \in \mathbb{N}^+$, p is* m-monotone *if for all $r_2 \ge r_1 + m$, $p(r_1) \ge p(r_2)$.*

Weak monotonicity guarantees that precision is non-increasing for points ranked at least m apart. We will use this to show that it is sufficient to compute precision at only logarithmically many points.

Definition 3 (Strong Monotonicity)**.** *For $\Delta, m \in \mathbb{N}^+$ s.t. $m \ge \Delta$, p is* (Δ, m)-monotone *if for all $r_2 \ge r_1 + m$,*

$$p_\Delta(r_1) \;\ge\; \frac{\sum_{r=r_1+1}^{r_2} p(r)}{r_2 - r_1} \;\ge\; p_\Delta(r_2)$$

Strong monotonicity says that for every large enough rank interval $[r_1, r_2]$, the Δ-precisions at the two ends of the interval bound the average precision across the interval. We will use this property to bound the actual precision function p by functions of local precision p_Δ.

We observe that $p(r)$ is simply $p_r(r)$. Further, when r is a multiple of Δ, $p(r)$ can be decomposed as $\frac{\Delta}{r}\sum_{j=1}^{r/\Delta} p_\Delta(j\Delta)$. Computing all of these uniformly spaced r/Δ local precision terms is, however, still as expensive in terms of the required annotation as computing $p(r)$ directly. We will instead *approximately decompose* $p(r)$ using roughly only $\log_{1+\epsilon} r$ local precision terms that are spaced based on a geometrically increasing spread with geometric ratio $1+\epsilon$.

4.2 SETUP AND ASSUMPTIONS

Throughout this section, let $T = (t_1, t_2, \ldots, t_N)$ be the ranked output of an algorithm $\mathcal{A}$ with (unknown) true labels $v(t_i) \in \{0,1\}$. Let p be $\mathcal{A}$'s precision function. Let $\Delta \in \mathbb{N}^+$ and p_Δ be the corresponding local precision function (cf. Definition 1). Let $\epsilon \in (0,1], \tilde{r} \in \mathbb{N}$ be such that $\tilde{r} \ge \lceil\frac{\Delta+2}{\epsilon}\rceil, \ell = \lceil\log_{1+\epsilon}\tilde{r}\rceil, m = \lfloor\epsilon(1+\epsilon)^\ell - 1\rfloor, \gamma = 1+\epsilon+\frac{2+\epsilon}{m}$, and $L = \lfloor\log_{1+\epsilon}N\rfloor$.

Observe that $m > \epsilon(1+\epsilon)^\ell - 2 \ge \epsilon\tilde{r} - 2 \ge \Delta$. The notion of (Δ, m)-monotonicity can thus be applied. Our method will involve annotating the true labels v for all of the first (roughly) $\tilde{r}$ items in the ranked list, followed by fewer than $\Delta\log_{1+\epsilon}N$ annotations for the rest of the list, in order to guarantee a $\gamma(1+\epsilon)$-approximation of p.

Assumptions. The algorithms and results below rely on one or both of the following assumptions:

A.1 p is m-monotone for all $r \geq \tilde{r}$.

A.2 p is (Δ, m)-monotone for all $r \geq \tilde{r}$.

While one intuitively expects monotonicity to hold (for large enough m and Δ) for any dataset produced by a well-designed prediction system, how often this happens in practice is an empirical question. We provide two pieces of support for it in Section 5: (a) visual support via the monotonically non-increasing ground truth points in Figures 2 and 3; and (b) quantitative support via the success of our method on two large and diverse datasets.

We noticed that the assumption fails near the 15K point in Figure 2, where the black ground truth curve starts to rise. Our estimate, as expected, deviates here a little, but then quickly regains accuracy as one moves to the right.

4.3 LOGARITHMIC EVALUATIONS

We start with a general result that evaluating p at $\mathrm{O}(\log_{1+\epsilon} N)$ points is sufficient to obtain a $(1+\epsilon)$-approximation of the entire precision function p. The idea is to compute p at the following geometrically spaced points, where the spacing is determined by ϵ:

Definition 4. *For $j \in \mathbb{N}^+$ and $\epsilon > 0$, define $g_{\epsilon,j} = \lceil (1+\epsilon)^j \rceil$.*

For brevity, when ϵ is clear from the context, we write g_j to mean $g_{\epsilon,j}$. Observe that $g_{j+1} - g_j \geq (1+\epsilon)^{j+1} - (1+\epsilon)^j - 1 = \epsilon(1+\epsilon)^j - 1$. When $(1+\epsilon)^j \geq \frac{m+1}{\epsilon}$, we thus have $g_{j+1} - g_j \geq m$. If we assume p is m-monotone for large enough r, we can show that evaluating p at only roughly $\log_{1+\epsilon} N$ points is sufficient to obtain a $(1+\epsilon)$-approximation of the entire precision function p. Formally, we define a step-function approximation:

Definition 5. *Let N, ϵ, ℓ be as in Section 4.2 and $f : [N] \to [0,1]$. Then $step^f_{\epsilon,\ell} : [N] \to [0,1]$ is defined as:*

$$step^f_{\epsilon,\ell}(r) = \begin{cases} f(r) & \text{if } r \leq g_\ell \\ f(g_j) \text{ for } j = \lfloor \log_{1+\epsilon} r \rfloor & \text{otherwise} \end{cases}$$

In other words, $step^f_{\epsilon,\ell}(r)$ can be computed using $g_\ell + L - \ell$ evaluations of f, mirrors $f(r)$ for small r, and is a geometrically adjusting step function afterwards. The following theorem, whose proof is deferred to the Appendix, shows that under the weak monotonicity assumption, $step^p_{\epsilon,\ell}$ is a tight approximation of p.

Theorem 1. *Let $p, \epsilon, \tilde{r}, \ell, m, L$ be as in Section 4.2. If Assumption A.1 holds, then $step^p_{\epsilon,\ell}$ is a pointwise $(1+\epsilon)$-approximation of p and can be computed using evaluations of p at points $1, 2, \ldots, g_\ell, g_{\ell+1}, \ldots, g_L$.*

This result as such is not directly helpful when evaluations of p are costly, as computing p *exactly* even at a single point requires a linear number of true label annotations (e.g., computing $p(N)$ exactly requires N annotations). However, what the result shows is that *if we could efficiently compute point-estimates of p, we would need to do so at roughly only $\log_{1+\epsilon} N$ points*:

Corollary 1. *Let $p, \epsilon, \tilde{r}, \ell, m, L$ be as in Sec. 4.2 and $\beta \geq 1$. If Assumption A.1 holds and $q(r)$ is a β-approximation of $p(r)$ for $r \in \{1, 2, \ldots, g_\ell, g_{\ell+1}, \ldots, g_L\}$, then $step^q_{\epsilon,\ell}$ is a pointwise $\beta(1+\epsilon)$-approximation of p.*

4.4 EFFICIENT POINT ESTIMATES

We now consider various ways of efficiently (in terms of required label annotations) computing β-approximations q of p at the $\mathrm{O}(\log N)$ points $g_{\ell+1}, \ldots, g_L$, in order to then apply Corollary 1 to obtain a $\beta(1+\epsilon)$-approximation of the entire precision function p.

4.4.1 Stratified Sampling

A simple way is to estimate each of these $L - \ell$ points is to employ random sampling and bound correctness confidence via Hoeffding's inequality, Eqn. (2). Since each point estimate requires many samples, it is substantially more efficient (in terms of evaluations of p) to reuse samples obtained for g_k when evaluating p at g_{k+1}. The resulting lack of independence slightly weakens the probabilistic correctness guarantee (we instead use the union bound), but leads to significantly fewer samples. Specifically, only a $\frac{\epsilon}{1+\epsilon}$ fraction of the required samples need to be obtained afresh; the rest can be reused.

This is formalized in the *stratified sampling* mechanism described in Algorithm 1, where X_k is the set of random samples considered for the point g_k and S_k denotes the precision of X_k. Samples in X_{k+1} are a combination of reusing most samples from X_k and obtaining only a few new samples from the range $g_k + 1, \ldots, g_{k+1}$. For this algorithm, we can derive the following probabilistic correctness guarantees (see Appendix for proofs):

Lemma 1. *Let $T, v, p, \epsilon, \tilde{r}, \ell, L$ be as in Sec. 4.2. Let $\delta > 0$, p_{min} be the minimum value of p, and $\beta > 1$. Then, with probability at least $1 - \delta$, $q(r)$ in Algorithm 1 on input $(T, v, \epsilon, \tilde{r}, \delta, p_{min}, \beta)$ is a β-approximation of $p(r)$ for $r \in \{g_{\ell+1}, \ldots, g_L\}$.*

Putting this together with Corollary 1 and noting that $q(r) = p(r)$ in Algorithm 1 when $r \leq \ell$, we obtain:

Theorem 2 (Logarithmic Stratified Sampling)**.** *Let $T, v, p, \epsilon, \tilde{r}, \ell, m, L$ be as in Sec. 4.2. Let $\delta > 0$, p_{min} be the minimum value of p, and $\beta > 1$. If Assumption A.1 holds, then with probability at least $1 - \delta$, the output of*

Algorithm 1 Logarithmic Stratified Sampling for Approximating Precision Function

input $T = (t_1, t_2, \ldots, t_N), v, \epsilon, \tilde{r}, \delta, p_{\min}, \beta$
  $\ell = \lceil \log_{1+\epsilon} \tilde{r} \rceil$; $L = \lfloor \log_{1+\epsilon} N \rfloor$; $r = (1+\epsilon)^{\ell-1}$
  for $j = \ell$ to L **do**
    $r = r * (1+\epsilon)$; $g_j = \lceil r \rceil$
  end for
  $s = \lceil \frac{1}{2(\beta-1)^2 p_{\min}^2} \ln \frac{L-\ell}{\delta/2} \rceil$
  $X_\ell = s$ random samples from $\{1, \ldots, g_\ell\}$
  $S_\ell = \sum_{i \in X_\ell} v(t_i)$
  for $k = \ell$ to $L-1$ **do**
    $X_{k+1}^{(1)}$ = include each $i \in X_k$ independently with probability g_k / g_{k+1}
    $X_{k+1}^{(2)} = s - |X_{k+1}^{(1)}|$ random samples from $\{g_k + 1, \ldots, g_{k+1}\}$
    $X_{k+1} = X_{k+1}^{(1)} \cup X_{k+1}^{(2)}$
    $S_{k+1} = \sum_{i \in X_{k+1}} v(t_i)$
  end for
  $q(r) = \frac{1}{r} \sum_{i=1}^{r} v(t_i)$ for $r \in \{1, \ldots, g_\ell\}$
  $q(g_k) = S_k / s$ for $k \in \{\ell+1, \ldots, L\}$
  Compute $step_{\epsilon,\ell}^q$ using the above values of q
output $step_{\epsilon,\ell}^q$

Algorithm 1 on input $(T, v, \epsilon, \tilde{r}, \delta, p_{min}, \beta)$ *is a* $\beta(1+\epsilon)$*-approximation of* $p(r)$*. Further, Algorithm 1 evaluates* p *only at* $1, \ldots, g_\ell$ *and at* $\frac{\epsilon(L-\ell)}{2(\beta-1)^2(1+\epsilon)p_{min}^2} \ln \frac{L-\ell}{\delta/2}$ *points chosen randomly via stratified sampling.*

Note that since $L = \Theta(\log N)$, the stratified sampling approach requires $\Theta(\log N \log\log N)$ annotations.

4.4.2 New Approach: PAULA

While random sampling provides efficient single point estimates, using it to approximate the entire p with error probability $\leq \delta$ requires bounding the error probability of each point more strictly, namely, by $\frac{\delta}{L-\ell}$, and using the union bound over $L-\ell$ dependent random events.[2]

We develop a novel method called PAULA to obtain γ-approximations of p at points of interest by evaluating p at logarithmically many *deterministically chosen* points. Since there is no probabilistic confidence involved, γ-approximations of the points directly carry over to a tight approximation of entire p, without any loss.

To this end, we employ the notions of local precision and strong monotonicity introduced in Section 4.1. We begin by observing that for any $k \geq \ell$, $g_{k+1} - g_k \geq g_{\ell+1} - g_\ell \geq (1+\epsilon)^{\ell+1} - (1+\epsilon)^\ell - 1 = \epsilon(1+\epsilon)^\ell - 1 \geq m$. The strong monotonicity assumption A.2 thus implies $p_\Delta(g_{k+1}) \leq p_\Delta(g_k)$, i.e., the *local precision is monotonically non-increasing along the geometrically spaced points* of interest. We define our candidates for lower and upper bounds on p via telescopic sums of local precisions at these points, as follows. As we will shortly see (Lemma 3), these terms bound the *yield* of $\mathcal{A}$ at rank g_k, normalizing which by g_k generates bounds on $p(g_j)$.

Definition 6. *Let* $\Delta, \epsilon, \ell, L$ *be as in Section 4.2 and* $k \in \{\ell, \ldots, L\}$*. Then:*

$$Y_-(\epsilon, \ell, k, \Delta) = g_\ell \, p(g_\ell) + \sum_{j=\ell}^{k-1} (g_{j+1} - g_j) \, p_\Delta(g_{j+1})$$

$$Y_+(\epsilon, \ell, k, \Delta) = g_\ell \, p(g_\ell) + \sum_{j=\ell}^{k-1} (g_{j+1} - g_j) \, p_\Delta(g_j)$$

Algorithm 2 describes our precision function approximation method, PAULA. We abbreviate $Y_-(\epsilon, \ell, k, \Delta)$ and $Y_+(\epsilon, \ell, k, \Delta)$ as Y_-^k and Y_+^k, resp. The algorithm is, in fact, very simple: compute some derived parameters and then loop over k to compute Y_-^k and Y_+^k via Δ-precision computations as defined above.

A similar idea has been used previously by Ermon et al. (2013), but in a different context, namely discrete integration for probabilistic graphical models. Our more delicate analysis, motivated by a novel use case, extends their finding for $\epsilon = 1$ to any $\epsilon \in (0, 1]$.[3]

Lemma 2 captures the nature of the approximation provided by PAULA:

Lemma 2. *Let* $T, v, p, \Delta, \epsilon, \tilde{r}, \ell, m, \gamma, L$ *be as in Section 4.2. If Assumption A.2 holds and* $p(g_\ell) \geq p_\Delta(g_\ell)$*, then* $q^-(r)$ *and* $q^+(r)$ *in PAULA on input* $(T, v, \Delta, \epsilon, \tilde{r})$ *are pointwise* γ*-approximations of* $p(r)$ *from below and above, resp., for* $r \in \{g_{\ell+1}, \ldots, g_L\}$*.*

This follows from Lemmas 3 and 4 below, whose proof is deferred to the Appendix.

Lemma 3. *Let* $p, \Delta, \epsilon, \ell, m$ *be as in Section 4.2. If Assumption A.2 holds, then for any* $k \geq \ell$*:*

$$Y_-(\epsilon, \ell, k, \Delta) \;\leq\; g_k \, p(g_k) \;\leq\; Y_+(\epsilon, \ell, k, \Delta)$$

Lemma 4. *Let* $p, \Delta, \epsilon, \ell, m, \gamma$ *be as in Section 4.2. If Assumption A.2 holds and* $p(g_\ell) \geq p_\Delta(g_\ell)$*, then for all* $k \geq \ell$*:*

$$\gamma \cdot Y_-(\epsilon, \ell, k, \Delta) \;\geq\; Y_+(\epsilon, \ell, k, \Delta)$$

[2] Even if the samples are not shared and one obtains independent estimates for each point with error probability δ' each, the overall error probability is $1 - (1-\delta')^{L-\ell}$, which is very close to $\delta'(L-\ell)$ when δ' is small.

[3] Specifically, (a) the notion of (m, Δ)-monotonicity is irrelevant to that work, as the search space there always (implicitly) satisfies monotonicity; (b) Theorems 1 and 2 and Corollary 1 are unrelated to that work; and (c) the 2-approximation that arose there naturally from parity constraints is too loose to be useful as an approximation of the precision function.

Algorithm 2 PAULA for Approximating Precision Function

input $T = (t_1, t_2, \ldots, t_n), v, \Delta, \epsilon, \tilde{r}$
  $\ell = \lceil \log_{1+\epsilon} \tilde{r} \rceil$; $L = \lfloor \log_{1+\epsilon} n \rfloor$; $r = (1+\epsilon)^{\ell-1}$
  for $j = \ell$ to L **do**
    $r = r * (1+\epsilon)$; $g_j = \lceil r \rceil$
  end for
  $Y^{\ell}_{-} = Y^{\ell}_{+} = g_\ell \; p(g_\ell)$
  for $k = \ell$ to $L-1$ **do**
    $Y^{k+1}_{-} = Y^{k}_{-} + (g_{k+1} - g_k) \; p_\Delta(g_{k+1})$
    $Y^{k+1}_{+} = Y^{k}_{+} + (g_{k+1} - g_k) \; p_\Delta(g_k)$
  end for
  $q^-(r) = q^+(r) = \frac{1}{r} \sum_{i=1}^{r} v(t_i)$ for $r \in \{1, \ldots, \ell\}$
  $q^-(g_k) = Y^k_- / g_k$ for $k \in \{\ell+1, \ldots, L\}$
  $q^+(g_k) = Y^k_+ / g_k$ for $k \in \{\ell+1, \ldots, L\}$
  Compute $step^{q^-}_{\epsilon,\ell}$ and $step^{q^+}_{\epsilon,\ell}$ using the above values of q^- and q^+, resp.
output $(step^{q^-}_{\epsilon,\ell}, step^{q^+}_{\epsilon,\ell})$

The above lemmas, in fact, show that Y_- and Y_+ together provide a γ-approximation *jointly*, in that sense that if Y_- is far from p at some point, then Y_+ must be close to p at that point. For simplicity, Lemma 2 (and Theorem 3 to follow shortly) states a weaker version that each of Y_- and Y_+ is a γ-approximation of p, independently.

Combining Lemma 2 and Corollary 1, we obtain:

Theorem 3 (PAULA)**.** *Let $p, \Delta, \epsilon, \ell, m, \gamma, L$ be as in Section 4.2. If Assumptions A.1 and A.2 hold, and $p(g_\ell) \geq p_\Delta(g_\ell)$, then the output of PAULA on input $(T, v, \Delta, \epsilon, \tilde{r})$ provides pointwise $\gamma(1+\epsilon)$-approximations of p from below and above, resp. Further, PAULA uses evaluations of p only at $g_\ell + \Delta(L-\ell)$ deterministically chosen points.*

In typical use cases, g_ℓ can be taken to be a constant like 500 or 1000, after which the precision function stabilizes. With Δ being a constant (typically 50 to 100) and $L = \Theta(\log N)$, PAULA thus requires $\Theta(\log N)$ annotations.

4.4.3 Comparison With Random Sampling

PAULA starts with zero error in approximating $p(r)$ as long as $r \leq n_0 = \lceil \frac{\Delta+2}{\epsilon} \rceil$. As r increases beyond n_0, the error increases as we only take logarithmically many samples afterwards. Nonetheless, under the monotonicity assumption, the error remains provably bounded by $\gamma(1+\epsilon)$. This contrasts with the common random sampling approach discussed in Section 3, which is inaccurate in the beginning and starts providing a $(1+\alpha)$-approximation, where $\alpha = \gamma(1+\epsilon) - 1$, with confidence $1-\delta$ once $r \geq n_{\alpha,s}$ as defined in Eqn. (4).

As noted earlier, the random sampling approach requires $\Theta(\sqrt{N \log N})$ samples where as PAULA requires only $\Theta(\log N)$ samples. Further, given the same number s of true label annotations and the same desired approximation factor, *we find that $n_{\alpha,s}$ is often too large*. Asymptotically, since s scales as $\Theta(\log N)$ for PAULA, we see from Eqn. (4) that $n_{\alpha,s}$ scales as $\Theta(N)$ when using the same s. In other words, given the same number of samples, PAULA obtains an approximation that holds everywhere whereas the random sampling method is inaccurate at a linear fraction of the N points.

For concreteness, we provide a numeric example. Consider the Science tensor (to be described later) where $N = 217077$. Using $\epsilon = 0.03$, PAULA requires $s = 17392$ annotations to provide a $\gamma(1+\epsilon)$ approximation of p at every point. In contrast, the random sampling approach from Section 3 needs 47030 annotations to achieve the same approximation (with error probability $\delta = 0.05$ and $p(r) \approx 0.7$). Alternatively, given the same number s of samples, the random sampling baseline has a much larger error initially and provides a pointwise α-approximation of $p(r)$ only for $r \geq n_{\alpha,s} = 41056$.

4.4.4 Comparison With Stratified Sampling

Comparing the number of evaluations of p needed in Theorems 2 vs. 3 to achieve a $\gamma(1+\epsilon)$-approximation, we see that PAULA has an asymptotic advantage over stratified sampling: $\Theta(\log N)$ vs. $\Theta(\log N \log\log N)$.

From a practical perspective, the difference is in the factor multiplying the $L - \ell$ term in each, which is $\frac{\epsilon}{2(\gamma-1)^2(1+\epsilon)p^2_{\text{min}}} \log \frac{L-\ell}{\delta/2}$ for stratified sampling and simply Δ for PAULA. To illustrate how these terms compare in practice, consider a typical application with parameter values $\delta = 0.05$ (i.e., 95% correctness confidence), $\epsilon = 0.03$, and $p_{\text{min}} = 0.5$. If we assume p is stable after the first 1000 points (i.e., $\tilde{r} = 1000$), then $\ell = 234$. In this case, the first expression simplifies to roughly $5.8 \log 40 (L - 234)$. For a ranked list T of size $N = 10000$ (or 100000), we have $L = 311$ (389, resp.) and the expression evaluates to 46.6 (50.6, resp.). In contrast, Δ is independent of N and can often be taken safely to be somewhat larger than $1/\epsilon = 20$.

This difference can be small in practice. We note, however, that the stratified sampling approach considered here is substantially more efficient than the conventional one, as it exploits our finding (Theorem 1) that logarithmically many point-estimates are sufficient.

5 EXPERIMENTS

We begin by illustrating how the number of samples needed by various precision function estimation methods

scales as the desired approximation factor is varied.[4]

We will then evaluate the accuracy and effectiveness of PAULA on resources from two application domains: natural language processing and knowledge acquisition. In the first case, ground truth is available and we demonstrate that the bounds produced by PAULA are very close to it. In the second case, we compute bounds on the precision function for a larger resource and demonstrate that (a) the bounds are close to each other and (b) match a few independently generated point estimates.

5.1 NUMBER OF ANNOTATIONS NEEDED

For this experiment, we vary the desired approximation factor $(1+\alpha)$ and compute the number of annotations needed by various methods to achieve this level of accuracy in estimating the precision function. Specifically, we consider the random sampling baseline (Section 3), our logarithmic stratified sampling (Section 4.4.1), and our deterministic approach, PAULA (Section 4.4.2). For the last two methods, $1+\alpha$ is a function of ϵ, namely, $\gamma(1+\epsilon)$.

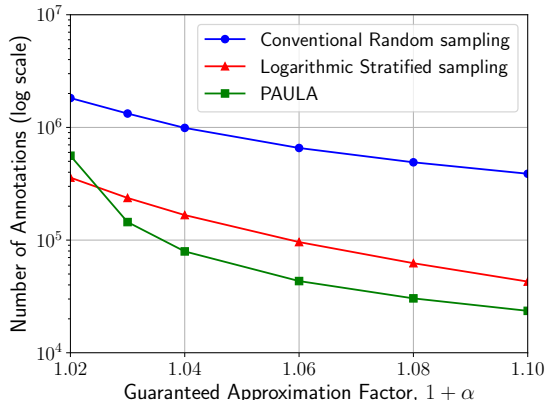

Figure 1: PAULA needs substantially fewer annotations to guarantee various levels of approximation of the precision function. $N = 10^7, p_{\min} = 0.5, \Delta = 100$.

Figure 1 depicts the number of annotations (in log scale) needed when N is 10M, $p_{\min}$ for the sampling based methods is taken to be 0.5, and Δ for PAULA is 100. The plot shows that *PAULA requires over an order of magnitude fewer annotations than the random sampling baseline* in order to achieve the same level of guaranteed accuracy. The logarithmic stratified sampling approach, which also exploits our Theorem 1, starts off as more efficient (in terms of annotations) than PAULA when α is small, but requires twice as many annotations when the approximation error is varied from 3% to 10%.

[4]Code and data available at http://allenai.org/software.

5.2 APPLICATION: NLP RESOURCES

We consider the Paraphrase Database PPDB (Ganitkevitch et al., 2013), a massive dataset for major NLP tasks such as paraphrase detection and entailment. Each paraphrase pair in PPDB 2.0 is associated with a validity score generated via supervised learning on 209 features (Pavlick et al., 2015). It also includes entailment relations, word embedding similarities, and style annotations. PPDB 2.0 can be viewed as a list of 169M items, ranked based on the trained paraphrase score.

In the first experiment, our goal is to demonstrate that the bounds provided by PAULA are close to the ground truth. To this end, we consider a subset of PPDB for which Pavlick et al. (2015) provide crowdsourced annotations on a 5-point scale (1-5) using five annotators per phrase pair.[5] If the average human judgment for a phrase pair t_i is at least 3, we consider $v(t_i) = 1$; otherwise, $v(t_i) = 0$.

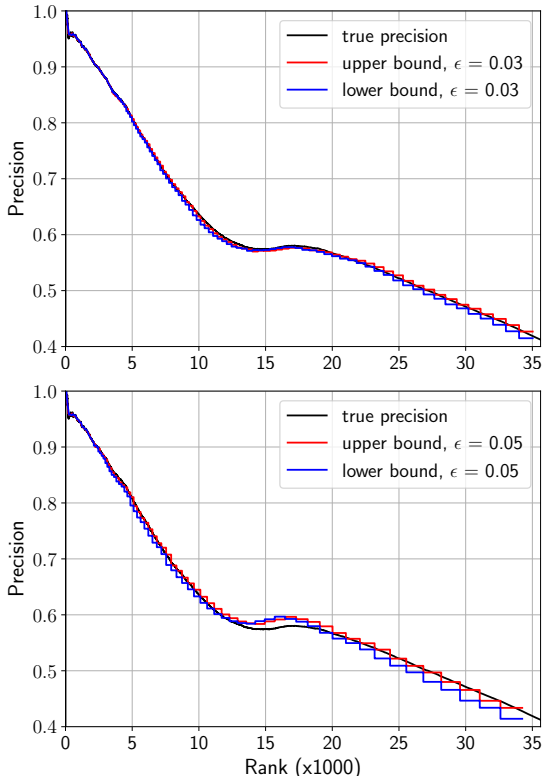

Figure 2: Bounds provided by PAULA with $\Delta = 100$ are very close to the ground truth on a fully annotated subset of PPDB 2.0 of size 36K. Top: $\epsilon = 0.03$, 11K annotations. Bottom: $\epsilon = 0.05$, 8K annotations.

Figure 2 shows the ground truth precision function p (black line) for this dataset of $N = 35615$ items, along

[5]http://www.seas.upenn.edu/∼nlp/resources/ppdb-2.0-human-labels.tgz

with step functions corresponding to PAULA's upper (red) and lower (blue) bounds when $\epsilon = 0.03$ (top) and $\epsilon = 0.05$ (bottom), using $\Delta = 100$. The bounds are extremely tight in practice, even though they were obtained using only 11292 and 7822 annotations, resp.

The random baseline can provide a good estimate as well, but is generally less accurate in the earlier part of the curve and resulted in substantial random fluctuations, as also suggested by the error bounds in Figure 1 and the numerical examples discussed in Section 4.4.3.

Finally, we note that using the same setting for ϵ and Δ as in the upper plot would require only 40K annotations to generate a precision function curve for *entire* PPDB 2.0 containing $N = 169$M items.

5.3 APPLICATION: KNOWLEDGE BASES

As a second application, we consider evaluating the quality of knowledge bases (KBs) which store facts in a structured manner, e.g., in form of *(entity, relation, entity)* triples such as *(EiffelTower, locatedIn, Paris)* or *(butterfly, pollinate, flower)*. Knowledge acquisition is the task of extracting such knowledge from various unstructured resources, such as the Web, while knowledge completion is the task of inferring more facts given a partial KB. In both these tasks, machine learning is commonly employed to model the entities and relations, and score various combinations of them as candidate facts. The triples that score higher are considered more reliable.

In this second experiment, our goal is to demonstrate that PAULA can scale to large resources while still producing upper and lower bounds that are very close to each other and to point estimates obtained via random sampling.

To this end, we consider Science KB (Dalvi et al., 2017), a dataset with facts about elementary science, along with corresponding confidence scores.[6] The KB has $N = 217076$ triples, making it prohibitively expensive to assess its quality by analyzing all triples.

Figure 3 shows the precision function bounds produced by PAULA, using the same setup as earlier ($\epsilon = 0.03, \Delta = 100$), which led to 17392 annotations. Due to the lack of ground truth for this dataset, we compute three point estimates independently by drawing 2000 uniform samples for each and using Eqn. (3) to compute a confidence interval. We observe that the upper (red) and lower (blue) bounds are very close to each other for the entire dataset, and also near the independently obtained point estimates.

[6] Available at http://data.allenai.org/tuple-kb as Aristo Tuple KB v4 (March 2017).

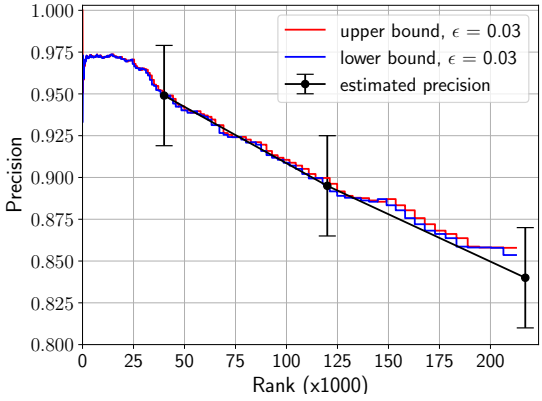

Figure 3: Bounds provided by PAULA with $\epsilon = 0.03, \Delta = 100$ using 17K annotations for Science KB.

6 CONCLUSION

This is the first work, to our knowledge, that provides a practical way of cost-effectively obtaining guaranteed approximations of the precision function, and associated metrics such as the PR curve, for massive datasets. This is particularly suitable for massive resources generated by today's large scale machine learning algorithms, when assessing individual data items for correctness requires human annotation. While these items are generally associated with a correctness score, and can thus be viewed as a ranked list, naïvely computing the PR curve of this list is prohibitively expensive due to the human annotation step. Consequently, these datasets often do not come with an associated precision-recall analysis, leaving their suitability for downstream applications unclear.

Our method, PAULA, addresses this limitation via tight approximations of the precision function with only logarithmically many annotations. PAULA leverages the fact that the precision function of reliable prediction systems is essentially monotonically non-increasing.[7] Our analysis reveals that sampling techniques require many more annotations (both asymptotically and for typical use cases) to guarantee the same level of approximation. Experiments on two large datasets demonstrate the accuracy of our method and its scalability.

Acknowledgments

The authors would like to thank Peter Clark, Oren Etzioni, Luke Zettlemoyer, and the anonymous reviewers for their valuable feedback, and Ellie Pavlick for sharing crowdsourced annotations for a subset of PPDB 2.0.

[7] For poorly designed prediction systems whose score does not correlate well with correctness, the monotonicity assumption is violated and PAULA's theoretical guarantees do not apply. PAULA can still be used to obtain a cost-effective estimate.

References

J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *SIGIR*, 2006.

B. Carterette, J. Allan, and R. K. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR*, 2006.

B. Dalvi, N. Tandon, and P. Clark. Domain-targeted, high precision knowledge extraction. *TACL*, 2017.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML*, 2006.

S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, 2013.

T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

J. Ganitkevitch, B. V. Durme, and C. Callison-Burch. PPDB: The paraphrase database. In *HLT-NAACL*, 2013.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

E. Kanoulas. A short survey on online and offline methods for search quality evaluation. In *RuSSIR*, 2015.

M. Majnik and Z. Bosnic. Roc analysis of classifiers in machine learning: A survey. *Intell. Data Anal.*, 17: 531–558, 2013.

T. M. Mitchell, W. W. Cohen, E. R. Hruschka, P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. T. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, 2012.

E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*, 2015.

T. Schnabel, A. Swaminathan, P. I. Frazier, and T. Joachims. Unbiased comparative evaluation of ranking functions. In *ICTIR*, 2016.

E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *CIKM*, 2006.

E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *SIGIR*, 2008.