RankDCG: Rank-Ordering Evaluation Measure

Denys Katerenchuk, Andrew Rosenberg
The Graduate Center, New York, USA
365 Fifth Avenue, Room 4319, New York, NY 10016
CUNY Queens College, New York, USA
65-30 Kissena Boulevard, Room A-202, Flushing, NY 11367
dkaterenchuk@gradcenter.cuny.edu, andrew@cs.qc.cuny.edu

Abstract

Ranking is used for a wide array of problems, most notably information retrieval (search). Kendall's τ, Average Precision, and nDCG are a few popular approaches to the evaluation of ranking. When dealing with problems such as user ranking or recommendation systems, all these measures suffer from various problems, including the inability to deal with elements of the same rank, inconsistent and ambiguous lower bound scores, and an inappropriate cost function. We propose a new measure, a modification of the popular nDCG algorithm, named rankDCG, that addresses these problems. We provide a number of criteria for any effective ranking algorithm and show that only rankDCG satisfies them all. Results are presented on constructed and real data sets. We release a publicly available rankDCG evaluation package.

Keywords: RankDCG, Rank Measure, Ranking, Ordering, Evaluation

1. Introduction

Every algorithm needs to be assessed to determine its performance. No single measure can be applied to all problems. If we consider a single area of computer science natural language processing (NLP), each problem requires specific evaluation. For example, for a simple classification task, accuracy is an intuitive and useful measure (Dumais et al., 1998; Katerenchuk et al., 2014). For named entity recognition and other "detection" tasks with a relatively small percent of relevant items (Tjong Kim Sang and De Meulder, 2003; Katerenchuk and Rosenberg, 2014), F-measure (Rijsbergen, 1979) is best suited. Correlation measures such as Pearson's r (Pearson, 1895) and Spearman's ρ (Spearman, 1904) are used to find relationships between entities (Strapparava and Mihalcea, 2008; Schuller et al.,). Kendall's τ (Kendall, 1938), Average Precision (AP) (Zhu, 2004), Mean Average Precision (MAP) and Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2002) are all used in information retrieval (IR) and ranking type of problems (Lapata, 2006; Philbin et al., 2007; Järvelin and Kekäläinen, 2000).

Despite a large number of different ranking measures, there are still problems that cannot be appropriately evaluated. One particular problem is to rank discrete value elements with multiple ties of the same rank and a skewed rank distribution. This type often arises in a number of ranking problems such as information retrieval or search. While some measures address parts of this problem, none exists to solve them all.

In this paper, we propose a new measure to deal with rank-ordering problems. We start with defining the problem and criteria that need to be satisfied in section 2. In Section 3 we give a short survey of available evaluation measures. In Section 4. we propose rankDCG, an improved evaluation measure and evaluate its performance in Section 5. We sum up our findings and conclude our work in Section 6.

2. Ordering

The problem of ordering is well known. It involves tasks such as internet search. The objectives are to find and order information from a near infinite set of data, namely web pages. Formally it can be defined as follows:

> Given a list of elements A = $[x_1, x_2, x_3, ..., x_n]$, the objective is to find list B = $[x|x \in A, f(x) > 0]$, where $f(x)$ is a relevance function that returns a rank that is higher or equal to 0. Often additional objective is applied such as B = $[f(x_1) > f(x_2) > f(x_3) > ... > f(x_m)]$ where m is a number of relevant document with $f(x) > 0$ and $m \leq n$.

In order to evaluate this problem, we require a comparison between two lists: the reference and the hypothesis. Relevance and ordering are the two prime factors that need to be considered. Because most measures were designed for IR tasks, the relevance of elements plays a crucial role in determining the evaluation score lower bound. In other words, if all elements in the hypothesis list are irrelevant, the score should be equal to 0 or some other lower bound.

In this paper, we consider an ordering problem that often appears in real word problems such as recommendation systems and user ranking. These tasks might appear identical to the web search problem. However, there is a number of distinct characteristics. Firstly, each element is relevant, meaning no irrelevant entities. Second, the element ranks are discrete values. Third, the rank values are not unique. In other words, there are many elements of the same rank referred to as multiple ties. Lastly, the elements might not follow the normal distribution of rank values. This case is also common to web search where only very few top results are relevant and the majority are somewhat related or not relevant to the query at all. The problem can be formally described in the following way:

Given a list of elements A = $[x_1, x_2, x_3, ..., x_n]$, the objective is to find list B = $[f(x_1) \geq f(x_2) \geq f(x_3) \geq ... \geq f(x_n)]$ where $f(x)$ is a rank function that returns rank r with $r \in N$ and n is a number of elements.

All conventional evaluation measures have a number of shortcomings while evaluating this type of problem. For this reason, we propose a set of criteria that an evaluation measure. This measure needs to address the following objectives:

1. correctly work with multiple ties
2. address non-normal rank value distribution
3. emphasize correct ordering of high rank elements
4. produce consistent and meaningful scoring range

In the next section, we will survey available algorithms and highlight some drawbacks of the most common rank evaluation measures.

3. Evaluation Measures Survey

Multiple rank-ordering evaluation metric algorithms exist in the field of information retrieval (IR). However, none of them is appropriate for the task described in the previous chapter. Keeping in mind the specifics of the problem, we survey various metrics, analyze their performance, and underline drawbacks.

3.1. F-measure

F-measure or F-score (Rijsbergen, 1979) is a common evaluation measure that is used to measure IE algorithms such as search (Peng and McCallum, 2006). This measure is defined as follows:

$$F = 2 * \frac{p * r}{p + r},$$

where p - precision and r - recall

F-measure takes into account precision and recall. Precision measures the portion of retrieved elements that are relevant. Recall measures the portion of relevant elements that were discovered. However, this measure is not appropriate for problems where all elements are relevant. In addition, this measure does not take into consideration different ranks. F-measure only evaluates the number of relevant elements; therefore, it is not suitable for a rank-ordering evaluation.

3.2. Average Precision and Mean Average Precision

Average Precision (AP) (Zhu, 2004) is a measure that is designed to evaluate IR algorithms. AP can deal with non-normal rank distribution, where the number of elements of some rank is dominant. AP measures precision at each element, multiplies the change in recall from the previous step, and averages over all list elements. There exists a variation of AP that takes into consideration only the first k elements. However, since we are concerned with a ranking of all elements, we will not focus on this variant. The formula to calculate the AP is the following:

$$AP = \sum_{k=1}^{n} P(k) * \Delta R(k)$$

where $P(k)$ = precision@k and $\Delta R(k)$ = $|recall(k-1) - recall(k)|$.

Researchers often use mean average precision (MAP) (liu, 2009), which is defined as the mean of AP over multiple information retrieval lists.

$$MAP = \frac{\sum_{q \in Q} AP(q)}{|Q|},$$

where Q = a set of ordering problems and q = a single evaluation instance.

Both AP and MAP measures have been designed to evaluate rank-ordering problems. The measures, however, assume no ties among ranks which manifests in inconsistent lower bounds. Furthermore, these measures evaluate all rank values with equal cost. However, the problem described in Section 2 requires more emphasis on ordering of rare high-rank elements and less for low-rank elements since these elements are not as important and often overrepresented. This creates a problem where misplacing a low-rank element can produce a low score, despite the fact that this element might not be very relevant to an otherwise good ordering result. More detail of this case can be found in Section 5.

3.3. Kendall's τ

Kendall's τ (Kendall, 1938) is a correlation measure. This measure is often used when evaluating rank-ordering results. The measure considers the number of element pairs in reference and hypothesis lists and checks whether the element positions correlate. The formal definition of Kendall's τ is shown below:

$$\tau = \frac{c - d}{\frac{1}{2}n(n-1)},$$

where c - a number of concordant (i.e. a correct relative ranking) pairs and d - a number of discordant (i.e. an incorrect relative ranking) pairs.

Kendall's τ is a popular choice for rank evaluation. Unfortunately, this measure also has some drawbacks. First of all, it does not explicitly deal with multiple ties and non-normal rank distribution. This will lead to a problem when an algorithm assigns the same (majority) rank value to all elements. Secondly, Kendall's τ does not produce a consistent lower bound score when the ranks follow a non-normal distribution. In addition, the score is produced by comparing the number of correlated elements and it does not emphasize rare high-rank elements. For these reasons, Kendall's τ is not the best choice to evaluate rank-ordering problems defined in Section 2.

3.4. Discounted Cumulative Gain

Among all evaluation measures, Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2002) has multiple advantageous characteristics to address a rank-ordering problem mentioned in the previous section. For this reason, it is often used in research (Lapata, 2006; Philbin et al., 2007; Järvelin and Kekäläinen, 2000). The main distinction of DCG from other measures is the ability to address non-normal rank distribution by assigning a higher cost to high-rank elements. This emphasizes the high-rank element identification. The formal definition of DCG is defined below:

$$DCG = \sum_{i=1}^{n} \frac{rel(x_i)}{log_2(i+1)},$$

where n - a number of elements and rel() - some relevance function of the i-th element in a given list.

For comparison across multiple tasks, a normalized variant of DCG, nDCG, is calculated in the following way:

$$nDCG = \frac{DCG}{IDCG},$$

where IDCG - represents the ideal DCG.

This evaluation also has its drawbacks. The first drawback is this evaluation metric was designed for information retrieval rather than ordering evaluation. This means that this measure considers the number of relevant and irrelevant documents. Since all elements in the rank-ordering task defined in Section 2. are relevant, the measure's lower bound is never equal to zero. As a result, the range of prediction is from 1, in the best case ordering, to some arbitrary number between 1 and 0. This factor makes results hard to understand because an nDCG score of 0.56 might be the worst case ordering. Another drawback is that the cost function puts too much stress on the high-rank element identification. The cost function was intentionally designed this way to bring more relevant search results closely to the top. However, the rank-ordering problem needs a relative function with respect to the rest of the elements. Lastly, standard DCG produces different cost based on the element positioning. For example a list [9,1,1] will have different costs for [1,9,1] and [1,1,9]. However, we contend that the two lists are equally wrong because the algorithm decided that element of rank 9 is rank 1. The permutations inside rank subgroup should not matter in the evaluation process.

4. Rank Discounted Cumulative Gain

In this section, we present a novel measure which we call rank discounted cumulative gain (rankDCG). This measure is a modified version of the popular nDCG algorithms. From Section 3., we can see that conventional evaluation measures fall short from addressing evaluation criteria. In particular, a good measure for rank-ordering problems needs to address the following:

1. multiple ties
2. non-normal rank value distribution
3. emphasis on high-rank elements
4. consistent scoring range

In order to demonstrate our algorithm, we start with constructing an example problem. The list L that is shown below, is ordered by rank values. In other words, each element represents an output from $rel(i)$, a relevance function. The rank values are discrete and the list contains multiple ties of elements with the same rank. In addition, the element rank distribution is non-normal.

$$L = [9_1, 4_2, 4_3, 2_4, 2_5, 2_6, 1_7, 1_8, 1_9, 1_{10}]$$

The first property rankDCG needs to address is non-normal rank distribution. From the Section 3., we saw that most rank measures, with the exception of DCG, do not have a way to distinguish between low-rank and high-rank elements in the scoring function. For this reason, we consider a number of cost functions that are similar to the DCG definition. We experiment with four different functions performed on list L and plot them in figures 1-4. The x-axis of each plot is the element order in list L and the y-axis is a cost generated from the experimental functions.

We start with an analysis of two cost functions: the standard DCG cost function and a modified version used in (Burges et al., 2005). From the Figures 1 and 2, we can see that both functions put more than half of their weight on the correct identification of the highest element. This can introduce bias toward finding the top-rank element rather than ordering. To address this issue, we design a function $rel'(i)$ that produces an element rank based on the number of unique element ranks in the list. The list L contains ten relative rank values, but only four unique values. We create a mapping function to assign a unique rank based on the rank subgroup. In other words, the top-rank element is equal to the size of the element rank set, $|\{L\}|$. Every following distinct element-rank will have its rank decreased by one. The results are plotted in Figure 3. In this case, given list L to the function $rel'(i)$ we get a corresponding list L' with the following ranks:

$$L' = [4_1, 3_2, 3_3, 2_4, 2_5, 2_6, 1_7, 1_8, 1_9, 1_{10}]$$

In addition to the above modification, we modify the discounting factor in the denominator of the DGC formula. DCG's discounted factor relies on the position of each

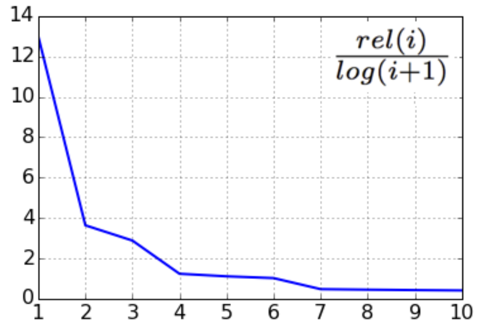

Figure 1: Cost function: $\frac{rel(i)}{log(i+1)}$

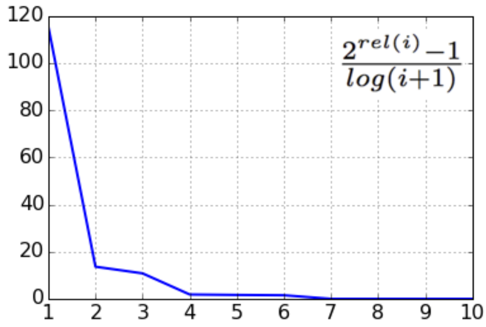

Figure 2: Cost function: $\frac{2^{rel(i)}-1}{log(i+1)}$

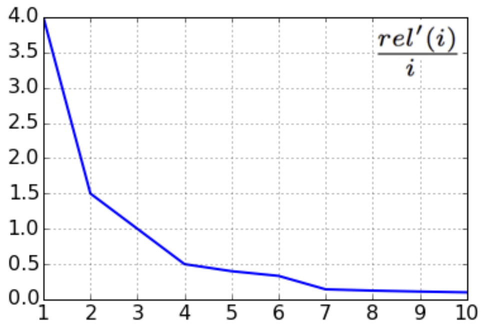

Figure 3: Cost function: $\frac{rel'(i)}{i}$

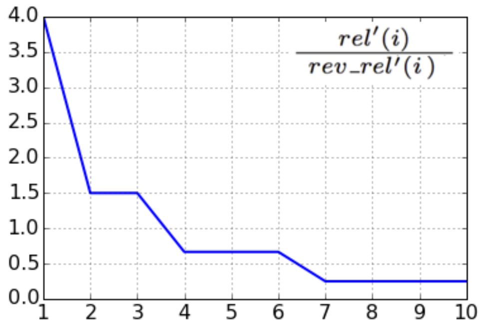

Figure 4: Cost function: $\frac{rel'(i)}{rev\_rel'(i)}$

element, and this implies that the last four value of L' list will produce different costs. Instead of using the elements' position, we find that reversed mapping order of $rel'()$ function works the best for discounted factor. The mapping between elements in L' and the discounted factors are represented in list D and the final cost function is shown in Figure 4. This discounted factor creates a step-wise function that eliminates a chance of getting a different score from permutations inside element subgroups.

$$D = [1_1, 2_2, 2_3, 3_4, 3_5, 3_6, 4_7, 4_8, 4_9, 4_{10}]$$

At this point the cost function is the following:

$$DCG' = \sum_{i=1}^{n} \frac{rel'(i)}{rev\_rel'(i)},$$

where n - a number of elements, $rel'(i)$ - cost function that takes L and creates L' and $rev\_rel'(i)$ - reversed rel'(i) function that takes L and creates discounted factor for each element that is shown in list D.

At last, we normalize DCG' to create a meaningful and consistent lower bound. The final normalized version of rankDGC is defined below:

$$rankDCG = \frac{max(DCG') - DCG'}{max(DCG') - min(DCG')}$$

Python implementation of rankDCG is available for download at our website (http://speech.cs.qc.cuny.edu).

5. Experiments

In this section, we show that rankDCG satisfies all the objectives and outperforms conventional rank-ordering measures on the constructed and the real data. The specified objectives are the following:

1. correctly work with multiple ties
2. address non-normal rank value distribution
3. emphasize correct ordering of high-rank elements
4. produce consistent and meaningful scoring range

5.1. Constructed data

We evaluate the behavior of rankDCG in seven possible scenarios: 1) perfect ordering, 2) misplacing low-rank elements, 3) misplacing a high-rank element with a medium rank element, 4) and 5) misplacing high and low rank elements, and 6) the worst case (reversed ordering). All the experiments are conducted on the list L = [9, 4, 4, 2, 2, 2, 1, 1, 1, 1] defined in the previous section and hypothesis list in Table 1. The results can be found in Table 1.

From the table, you can see that only rankDCG satisfies our criteria. Starting with the objective 1, we can see that only Kendall's τ and rankDCG address it properly. The score of comparing reference list L and lists 4 and 5 from the table produce the same score. This fact brings robustness

#	Hypothesis List	Kendall's τ	AveP	nDCG	rankDCG
1	[9, 4, 4, 2, 2, 2, 1, 1, 1, 1]	1.0	1.0	1.0	1.0
2	[9, 4, 4, 2, 2, 1, 2, 1, 1, 1]	0.8	0.887	0.998	0.975
3	[4, 4, 2, 9, 2, 2, 1, 1, 1, 1]	0.742	0.454	0.825	0.65
4	[1, 4, 4, 2, 2, 2, 9, 1, 1, 1]	0.285	0.659	0.688	0.325
5	[1, 4, 4, 2, 2, 2, 1, 1, 1, 9]	0.285	0.697	0.667	0.325
6	[1, 1, 1, 1, 2, 2, 2, 4, 4, 9]	-0.8	0.149	0.571	0.0

Table 1:

to possible element permutations inside a subgroup of elements with the same rank.

The second and third objectives are the ability to deal with a non-normal distribution and emphasize correct ordering of rare, top-rank, and elements. RankDCG produces the most accurate cost function. This can be observed by comparing reference list L to lists 2, 3 and 4 in the table. In the case of the comparison with list 2, most measures produced reasonable results. NDCG puts little cost on misidentifying low-rank elements. This score follows my rankDCG, with AveP and τ being the harshest score of 0.8 for miss-ordering low-rank element. On the other hand, τ puts very little cost on misplacing the top element (0.742). This fact makes high-rank element ordering of a lesser importance. If we look at case 4, we can see that AveP gives a higher score of 0.650 for placing the top-rank element into the lowest-rank group, compared to 0.454 score for placing the same element into a better subgroup. Among all score variations, rankDCG fits right in the middle with the scoring cost function and produces a linear score decrease with worse ordering case.

Finally, due to the initial application of the surveyed measures in the IR area, none of the measures satisfies the lower bound requirement. This can be observed in case 6, which is the worst case ordering, where all measures produce scores that are difficult to understand. The score from τ and AveP show that the results are not good, but not the worst possible case. NDCG's score can be interpreted as a good result by a person not familiar with the measure or the task. RankDCG is the only measure that produces a comprehensive worst case score.

5.2. Real data

One real world problem where common measures fall short is user ranking. This task involves ranking users according to their community rank. We are looking at the Reddit website (www.reddit.com). Reddit is a website where users create posts on different topics or share resources such as pictures, videos, or links to other resources. Users can participate in discussions through creating threads of comments. Each comment can earn comment karma, which is Reddit's form of approval. We consider data from politics subreddit. We rank users from five randomly chosen subreddits that contain at least one-hundred comments. On average, each subreddit contains 129.8 users. Using NLP algorithms, we analyze the comments and predict user rank (karma index) based on text analysis. This problem is very challenging and the results are far from perfect. However, to demonstrate shortcomings of popular rank-measures we create four tests: 1) we limit the data and produce a bad ranking prediction using limited part-of-speech analysis, 2) a slightly better rank predictions using LIWC word list (Pennebaker et al., 2001), 3) further improved ranker using n-gram approach, and 4) the perfect prediction, comparing the reference with itself. The results can be found in Table 2.

#	Kendall's τ	AveP	nDCG	rankDCG
1	nan	0.79	0.883	0.0
2	0.197	0.668	1.188	0.32
3	0.136	0.585	1.318	0.347
4	0.5	1	1	1

Table 2:

From the Table 2, we can see a few interesting cases. In the first case, the rank-ordering algorithm based on limited data outputs the majority class. As a result, we can see that Kendall's τ cannot deal with this case while AveP and nDCG returned seemingly good results. In the second case, with a slightly better ranking model, all measures show improvement. nDCG returns a score higher than 1 because the algorithms overpredict high ranks. The third case, a better model, perceives the results as worse than the results from the second case by Kendall's τ and AveP. nDCG, on the other hand, produces a higher than 1 score. In the last fourth case, the perfect ordering, all measures, with the exception of Kendall's τ, produce correct scoring. After considering all the cases, we can see that only rankDCG shows consistent evaluation scores with a gradual improvement of the algorithm and meaningful lower and upper bounds.

6. Conclusion

In this paper, we present rankDCG, a rank-ordering evaluation measure. RankDCG is a modification of popular nDCG algorithm that addresses some shortcomings of common evaluation measures. While there is a number of popular evaluation measures available, we show that they cannot properly evaluate ranking problems with discrete values, multiple ties, and nonlinear rank distribution. In this work, we define criteria that a good evaluation measure needs to submit and show that among popular measures, only rankDCG satisfies it. We release the rankDCG evaluation package to the public as a part of this work and make it available on our website [1].

[1] http://speech.cs.qc.cuny.edu

7. Bibliographical References

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA. ACM.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 148–155, New York, NY, USA. ACM.

Järvelin, K. and Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October.

Katerenchuk, D. and Rosenberg, A. (2014). Improving named entity recognition with prosodic features. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Katerenchuk, D., Brizan, D. G., and Rosenberg, A. (2014). "was that your mother on the phone?": Classifying interpersonal relationships between dialog participants with lexical and acoustic properties. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, 30(1/2):81–93, June.

Lapata, M. (2006). Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

(2009). Mean Average Precision. In LING LIU et al., editors, *Encyclopedia of Database Systems*, pages 1703–1703. Springer US.

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, pages 240–242.

Peng, F. and McCallum, A. (2006). Information extraction from research papers using conditional random fields. *Inf. Process. Manage.*, 42(4):963–979, July.

Pennebaker, J. W., Francis, M. E., and Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Schuller, B., Steidl, S., Batliner, A., Hantke, S., Hönig, F., Orozco-Arroyave, J. R., Nöth, E., Zhang, Y., and Weninger, F.). The interspeech 2015 computational paralinguistics challenge: Nativeness, parkinson's & eating condition.

Spearman, C. (1904). "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):pp. 201–292.

Strapparava, C. and Mihalcea, R. (2008). Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560. ACM.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhu, M. (2004). Recall, precision and average precision.