Inducing Controlled Error over Variable Length Ranked Lists

Laurence A.F. Park and Glenn Stone

School of Computing, Engineering and Mathematics,
University of Western Sydney, Australia
{l.park,g.stone}@uws.edu.au
http://www.scem.uws.edu.au/~lapark

Abstract. When examining the robustness of systems that take ranked lists as input, we can induce noise, measured in terms of Kendall's tau rank correlation, by applying a set number of random adjacent transpositions. The set number of random transpositions ensures that any ranked lists, induced with this noise, has a specific expected Kendall's tau. However, if we have ranked lists of varying length, it is not clear how many random transpositions we must apply to each list to ensure that we obtain a consistent *expected* Kendall's tau across the collection. In this article we investigate how to compute the number of random adjacent transpositions required to obtain an expected Kendall's tau for a given list length, and find that it is infeasible to compute for lists of length more than 9. We also investigate an alternate and more efficient method of inducing noise in ranked lists called Gaussian Perturbation. We show that using this method, we can compute the parameters required to induce a consistent level of noise for lists of length 10^7 in just over six minutes. We also provide an approximate solution to provide results in less than 10^{-5} seconds.

1 Introduction

The robustness of a modeling or prediction system is defined as the system's ability to handle noise or error applied to its input, and operate within certain limits. For example, if we have a classification system that predicts a state, given a set of observations, we can measure its accuracy by examining how many of its predictions are correct. We can then measure the robustness of the system by applying a specified level of random noise to the observations, and then examine its change in accuracy. Of course, if we increased the level of the added noise, we would expect the accuracy of the system to decrease, but a more robust system would provide a slower decrease.

To determine the robustness level of a system, we must often perform a simulation experiment, where we can control the noise. Noise is usually randomly sampled from a predefined distribution with carefully chosen parameters to ensure the noise is consistent for the experiment. For example, if our observations are elements of the real number set, we may generate noise using a Normal distribution with zero mean and variance of one. If our observations are frequency values, we may generate noise using a Poisson distribution with mean 1. Further experiments can then be performed by adjusting the variance of the noise distribution and measuring the change in accuracy.

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 259–270, 2014.

Many systems, such as collaborative filtering systems, meta search engines, query expansion systems, and rank aggregation systems require a set of ranked items as input.

Therefore, to measure the robustness of such systems, we can randomly permute the lists to obtain a given *expected* Kendall's τ between the permuted and unpermuted lists. We can achieve this by performing a set number of randomly chosen adjacent transpositions, as long as *all* of the list lengths are the same. If the lists lengths are not the same, it is not clear how many adjacent transpositions we should apply to obtain a given expected Kendall's τ across all lists.

In this article, we investigate how to compute the number of random adjacent transposition required to obtain a given expected τ for a given list length. We find that this is a very computationally expensive task. We also propose an alternate method of inducing error in ranked lists called Gaussian Perturbation. We show that its parameter is a function of the rank correlation reduction caused by the error, and therefore can be used to induce controlled noise in ranked lists. We provide the following contributions:

- An analysis of the relationship between the expected Kendall's τ, the number of random adjacent transpositions and the list length (Section 3.1),
- A novel ranked list noise induction method called Gaussian Perturbation that can be computed for larger lists in reasonable time, (Section 3.2),
- An approximate version of Gaussian Perturbation to compute the required parameters in minimum time (Section 3.3).

The article will proceed as follows: Section 2 reviews how we measure the robustness of a system, and examines the form of Kendall's τ. Section 3 examines the noise induced using random adjacent transpositions, introduces and analyses Gaussian Perturbation, and provides a faster approximation to Gaussian Perturbation.

2 Robustness Using Ranked Lists

To assess the robustness of a prediction system using simulation, we must define a method of inducing controlled noise into the observation space. In this section, we will examine how to measure robustness, given a noise distribution. We will then examine how to use Kendall's τ rank correlation to measure the induced noise.

2.1 Measuring System Robustness

A robust system is one that can function within a set of predefined limits in a noisy (error prone) environment. Therefore a robust classification system would be able to provide a certain level of accuracy, when making predictions based on observations, with a given level of noise. For example, if our observation space is the one dimensional real line $\mathbb{R}$, we might define an unknown true value as $a \in \mathbb{R}$, and an observation with added noise as $\hat{a} = a + \epsilon$ where $\epsilon \sim N(0, \sigma)$, a sample from a Normal distribution, with mean 0 and standard deviation σ.

In this situation the level of noise is controlled by the parameter σ. As the difference between the noisy and noise free observations increase, we would expect the system prediction accuracy to change, but we would expect a more robust system's behaviour to change less.

Analysis of robust system behaviour has provided us with strategies that allow prediction models to provide good accuracy using a wide range of observation input data. Some well known general strategies are to use cross validation when training a model [12] and including a regularisation term in the optimisation function (used in SVMs [13] and Lasso/Ridge regression [7]). Such systems introduce bias into the optimisation, in order to reduce the variance of the classification, and hence increase the robustness.

Similar methods can be used in clustering and dimension reduction. Compressive sampling has been used in image analysis [2], clustering [11] and outlier detection [1]. Each of these methods use l_1 regularisation to obtain sparse solutions to the dimension reduction, clustering, and outlier detection problems. These methods have also been applied to Principal Component Analysis [3], to obtain biased, but more robust principal components.

Now consider the observation space as the set of all ranked lists of multiple lengths. There are many systems that have an observation space of ranked lists, therefore it is important that we provide a method of measuring the robustness of this space. Systems using Collaborative filtering [9] use an observation space of ranked lists. These systems obtain ranked responses from the user community and aggregate them to assist in the decision making process. Query expansion [4] also requires ranked lists, where ranked documents are used to extract potential query terms. Meta search over multiple databases [8,6] obtain ranked results from various databases (e.g. airline travel prices, book prices, Web search results) and combines them to form a single result. Also, Rank aggregation systems [5,10] are used to combine multiple ranked lists into a single ranked list (e.g. results from tennis competitions to form an overall player ranking).

To test the robustness of methods applied in such systems, we need a method to induce noise into ranked lists. Furthermore, we must be able to control the *amount* of noise induced. Ranked lists contain ordinal numbers, therefore the error term ϵ would be the application of a random permutation of the list elements (clearly, simply adding an error variate to the *true* ranking will not be appropriate for ordinal numbers).

2.2 Measuring Error in Ranked Lists

The ranking of n items can be identified with an ordering of the integers $1, \ldots, n$. Thus the sample space for ranked lists can be considered S_n, the set of permutations of the integers 1 to n. We will talk about elements $\boldsymbol{x} = (x_1, \ldots, x_n)$ of S_n, where x_i is the rank of item i. Note that we cannot induce error on lists of length $n = 1$, therefore, we will only consider $n > 1$ in this article.

Given two rankings $\boldsymbol{x}$ and $\boldsymbol{y} \in S_n$, one way to measure the similarity of the two rankings is using Kendall's τ rank correlation. Kendall's τ is defined as:

$$\tau(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{1 \leq i < j \leq n} c_{ij} - d_{ij}}{n(n-1)/2}$$

where the sum is over the set of all possible pairs of items being ranked, and

$$c_{ij} = \begin{cases} 1, & \text{if } ((x_i < x_j) \text{ and } (y_i < y_j)) \text{ or } ((x_i > x_j) \text{ and } (y_i > y_j)) \\ 0, & \text{otherwise} \end{cases}$$

and $d_{ij} = 1 - c_{ij}$. A pair of items are concordant ($c_{ij} = 1$) if their ordering in each of the two lists matches, otherwise they are discordant ($d_{ij} = 1$). We can see that $\tau = 1$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (all items are concordant), implying perfect correlation. Also $\tau = -1$ if and only if $\boldsymbol{x} = \text{reverse}(\boldsymbol{y})$ (all items are discordant), implying perfect anti-correlation. The result of $\tau = 0$ implies no correlation between $\boldsymbol{x}$ and $\boldsymbol{y}$. Therefore, Kendall's τ can be simplified to:

$$\tau(\boldsymbol{x}, \boldsymbol{y}) = \frac{2\sum_{1 \leq i < j \leq n} c_{ij}}{n(n-1)/2} - 1$$

since $\sum_{1 \leq i < j \leq n} c_{ij} + d_{ij} = n(n-1)/2$.

We can use Kendall's τ to measure the noise induced in a ranked list. Given ranked lists $\boldsymbol{x}, \boldsymbol{y} \in S_n$, the level of noise between $\boldsymbol{x}$ and $\boldsymbol{y}$ is measured using $\tau(\boldsymbol{x}, \boldsymbol{y})$. The greater the value of τ, the lower the amount of noise.

Since Kendall's τ is a measure of correlation, it is comparable across lists of different lengths. This means that if we induce noise on a list of length n_1 and on another of length n_2, where the measure of noise using Kendall's τ is the same for both, then we have induced the same level of noise for both lists.

3 Inducing Controlled Noise in Ranked Lists

To examine the robustness of a system, we must examine how it performs when noise is introduced. If the system takes a set of ranked lists as input, we must induce controlled noise in the ranked lists. We showed that error in ranked lists can be measured using Kendall's τ, which is a function of the permutation required to remove error from the erroneous ranked list. Therefore, to induce controlled noise, we must perform controlled permutations.

Ideally, supposing the *truth* to be $\boldsymbol{x} \in S_n$, we would weight all elements $\boldsymbol{y} \in S_n$ such that the expected τ is as desired. We can then sample from the set S_n with probability proportional to the assigned weights. However, there are $n!$ elements in S_n and enumeration rapidly becomes impractical.

In this section, we examine two methods of sampling from S_n to obtain an expected τ; one controlled by the number of transpositions t, the other controlled by the standard deviation σ.

3.1 Using Adjacent Transpositions to Induce Controlled Noise

When measuring error using Kendall's τ, an obvious choice of inducing error is to perform adjacent transpositions. The set of adjacent transpositions is a generating set for the symmetric group, therefore we can obtain every possible ranking of n items using a finite sequence of adjacent transpositions.

To induce error in a ranked list $\boldsymbol{x}$ of length n, we randomly select x_i, where $i \in \{1, 2, \ldots, n-1\}$ and transpose it with the adjacent item x_{i+1}. By performing t random adjacent transpositions, we obtain a permuted list $\boldsymbol{y}$, which when compared to the original list $\boldsymbol{x}$, gives a value of τ. If we repeat this random process many times, we find that we obtain a distribution over τ, that is dependent on n and t.

To compute the expected Kendall's τ after t random adjacent transpositions, we first construct the probability transition matrix containing the probability of moving from one state to another in one adjacent transposition. The state of the list is the order of the items in the list. Therefore, if there are n items in the list, there are $n!$ possible states. The probability transition matrix will be of size $n! \times n!$, but each column will contain only $n-1$ nonzero elements (since for any list of n items, we can only move to $n-1$ other states using a single adjacent transposition). This gives us a probability transition matrix containing $n! \times (n-1)$ nonzero elements.

For example, if $n = 3$, we have the $n! = 6$ possible states $\boldsymbol{x}_1 = (1, 2, 3)$, $\boldsymbol{x}_2 = (1, 3, 2)$, $\boldsymbol{x}_3 = (3, 1, 2)$, $\boldsymbol{x}_4 = (3, 2, 1)$, $\boldsymbol{x}_5 = (2, 3, 1)$ and $\boldsymbol{x}_6 = (2, 1, 3)$ giving a probability transition matrix T with $n!(n-1) = 12$ nonzero values:

$$T = \begin{bmatrix} 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0.5 & 0 \end{bmatrix} \tag{1}$$

where $t_{i,j}$, the elements of T, contain the probability of moving from state j to state i. If we begin in state 1: $\boldsymbol{x}_1 = (1, 2, 3)$, our initial state probability vector is $\boldsymbol{p}_0 = [\ 1\ 0\ 0\ 0\ 0\ 0\ ]'$. By taking a random walk of length 1, we compute our new state probability as $\boldsymbol{p}_1 = T\boldsymbol{p}_0 = [\ 0\ 0.5\ 0\ 0\ 0\ 0.5\ ]'$. A random walk of length 2 is computed as $\boldsymbol{p}_1 = T\boldsymbol{p}_1 = T^2\boldsymbol{p}_0 = [\ 0.5\ 0\ 0.25\ 0\ 0.25\ 0\ ]'$. A random walk of length n gives us the state distribution $T^n\boldsymbol{p}_0$. Once we have the probability of each state after t random adjacent transpositions, we can compute the expected Kendall's τ using:

$$\mathbb{E}[\tau] = \sum_{i=1}^{n!} p_{t,i}\tau(\boldsymbol{x}_i, \boldsymbol{x}_1) \tag{2}$$

where $p_{t,i}$ is the ith element of $\boldsymbol{p}_t$. If we perform two random transpositions (a random walk of length 2), on a list of length $n = 3$, we can obtain a Kendall's τ of 0 or $-1/3$, but the expected Kendall's τ is:

$$\begin{aligned} \mathbb{E}[\tau] = {} & 0.5 \times 1 + 0 \times 1/3 + 0.25 \times (-1/3) \\ & + 0 \times (-1) + 0.25 \times (-1/3) + 0 \times 1/3 = 1/3 \end{aligned}$$

By representing the problem as a random walk on an undirected graph, we obtain additional information about its stationary distribution, being proportional to the degree of each vertex over an undirected graph. The degree of each vertex

Table 1. The number of nonzero elements (Nonzero) in the adjacent transposition adjacency matrix and the computation time required (Time) to compute the 50 expected values

List length	2	3	4	5	6	7	8	9
Nonzero	2	12	72	480	3600	30240	282240	2903040
Time (sec)	0.055	0.099	0.311	1.322	7.628	56.034	9.78 min	10.38 hr

over the set of permutations is equal $(n-1)$, therefore the stationary distribution is the Uniform. This implies that as the number of adjacent transpositions approaches infinity, each state is equally likely. Kendall's τ is symmetric about 0 over all permutation states, therefore the expected Kendall's τ approaches 0 as t approaches infinity for all lists of length $n > 1$.

Given the task of randomly sampling ranked lists of length n with a given expected τ error, it is not obvious how we should choose t. In fact there is no way to directly compute t, other than trial and error (set t and n and examine the associated expected τ). To achieve this task, we have provided Table 2 containing the computed expected τ values of lists of length 2 to 9, using 1 to 50 random adjacent transpositions. So given $\mathbb{E}[\tau] = 0.5952$, the table shows that we are required to perform 13 random adjacent transpositions for $n = 8$. If we induce error in lists of length 7 and 9, we find that 9 and 18 random adjacent transpositions will provide the wanted $\mathbb{E}[\tau]$ respectively.

We have also provided the computation time for each of the lists in Table 1. It is interesting to note how fast the number of nonzero elements and computation time grows as n increases. Based on the trend, we found that the time is increasing double exponentially, meaning that the expected τ for $n = 10$ would take approximately 100 days to compute. Clearly, it is not feasible to compute the expected τ using this method for lists of length 10 or more.

3.2 Using Gaussian Perturbation to Induce Controlled Noise

Rather than performing adjacent transpositions, controlled noise can be induced in ranked lists by perturbing the rank of the elements. Perturbation requires the introduction of a latent value for each item i as a sample from a Normal distribution $X_i \sim N(\mu = x_i, \sigma)$, with mean equal to its rank x_i and constant standard deviation σ.

Once we have sampled a value for each item, we generate the new rank, by ordering the latent values. An example of this process is shown in Figure 1. We can see in this example four items $(1, 2, 3, 4)$, each having a Normal distribution with equal standard deviation, centred on its rank. A sample from these four distributions gives $(1.2, 3.1, 2.7, 4.4)$, providing us with the noise induced ranked list $(1, 3, 2, 4)$. Using this method of noise induction, it is more likely that each item moves a smaller number of ranks than one item move a large number, which is the typical form of error seen in ranked lists.

Table 2. The expected Kendall's τ after t transpositions over lists is length 2 to 9

t	List length							
	2	3	4	5	6	7	8	9
1	-1.0000	0.3333	0.6667	0.8000	0.8667	0.9048	0.9286	0.9444
2	1.0000	0.3333	0.5556	0.7000	0.7867	0.8413	0.8776	0.9028
3	-1.0000	0.0000	0.4198	0.6125	0.7216	0.7901	0.8361	0.8685
4	1.0000	0.1667	0.3580	0.5469	0.6681	0.7469	0.8007	0.8389
5	-1.0000	-0.0833	0.2716	0.4883	0.6216	0.7091	0.7695	0.8127
6	1.0000	0.1250	0.2318	0.4395	0.5807	0.6754	0.7414	0.7890
7	-1.0000	-0.1042	0.1759	0.3955	0.5440	0.6449	0.7158	0.7673
8	1.0000	0.1146	0.1501	0.3571	0.5107	0.6170	0.6922	0.7472
9	-1.0000	-0.1094	0.1139	0.3223	0.4803	0.5911	0.6703	0.7285
10	1.0000	0.1120	0.0972	0.2913	0.4523	0.5672	0.6498	0.7109
11	-1.0000	-0.1107	0.0738	0.2633	0.4264	0.5447	0.6306	0.6943
12	1.0000	0.1113	0.0630	0.2381	0.4023	0.5237	0.6124	0.6786
13	-1.0000	-0.1110	0.0478	0.2153	0.3798	0.5038	0.5952	0.6637
14	1.0000	0.1112	0.0408	0.1947	0.3587	0.4850	0.5789	0.6494
15	-1.0000	-0.1111	0.0309	0.1761	0.3390	0.4672	0.5633	0.6358
16	1.0000	0.1111	0.0264	0.1593	0.3205	0.4503	0.5483	0.6227
17	-1.0000	-0.1111	0.0200	0.1441	0.3030	0.4342	0.5341	0.6102
18	1.0000	0.1111	0.0171	0.1303	0.2865	0.4188	0.5204	0.5981
19	-1.0000	-0.1111	0.0130	0.1179	0.2710	0.4041	0.5072	0.5864
20	1.0000	0.1111	0.0111	0.1066	0.2564	0.3900	0.4945	0.5752
21	-1.0000	-0.1111	0.0084	0.0964	0.2426	0.3765	0.4823	0.5643
22	1.0000	0.1111	0.0072	0.0872	0.2295	0.3636	0.4705	0.5538
23	-1.0000	-0.1111	0.0054	0.0789	0.2171	0.3511	0.4591	0.5436
24	1.0000	0.1111	0.0046	0.0714	0.2055	0.3391	0.4481	0.5337
25	-1.0000	-0.1111	0.0035	0.0645	0.1944	0.3276	0.4374	0.5241
26	1.0000	0.1111	0.0030	0.0584	0.1840	0.3166	0.4271	0.5148
27	-1.0000	-0.1111	0.0023	0.0528	0.1741	0.3059	0.4171	0.5057
28	1.0000	0.1111	0.0019	0.0478	0.1648	0.2956	0.4074	0.4968
29	-1.0000	-0.1111	0.0015	0.0432	0.1559	0.2857	0.3979	0.4882
30	1.0000	0.1111	0.0013	0.0391	0.1476	0.2761	0.3887	0.4798
31	-1.0000	-0.1111	0.0010	0.0353	0.1397	0.2669	0.3798	0.4716
32	1.0000	0.1111	0.0008	0.0320	0.1322	0.2580	0.3711	0.4636
33	-1.0000	-0.1111	0.0006	0.0289	0.1251	0.2494	0.3627	0.4558
34	1.0000	0.1111	0.0005	0.0262	0.1184	0.2411	0.3545	0.4482
35	-1.0000	-0.1111	0.0004	0.0237	0.1120	0.2331	0.3465	0.4407
36	1.0000	0.1111	0.0003	0.0214	0.1060	0.2254	0.3387	0.4335
37	-1.0000	-0.1111	0.0003	0.0194	0.1003	0.2179	0.3311	0.4263
38	1.0000	0.1111	0.0002	0.0175	0.0950	0.2106	0.3236	0.4193
39	-1.0000	-0.1111	0.0002	0.0158	0.0899	0.2037	0.3164	0.4125
40	1.0000	0.1111	0.0001	0.0143	0.0851	0.1969	0.3094	0.4058
41	-1.0000	-0.1111	0.0001	0.0130	0.0805	0.1904	0.3025	0.3993
42	1.0000	0.1111	0.0001	0.0117	0.0762	0.1841	0.2958	0.3928
43	-1.0000	-0.1111	0.0001	0.0106	0.0721	0.1780	0.2892	0.3865
44	1.0000	0.1111	0.0001	0.0096	0.0682	0.1721	0.2828	0.3804
45	-1.0000	-0.1111	0.0000	0.0087	0.0646	0.1664	0.2766	0.3743
46	1.0000	0.1111	0.0000	0.0078	0.0611	0.1609	0.2705	0.3684
47	-1.0000	-0.1111	0.0000	0.0071	0.0578	0.1556	0.2645	0.3626
48	1.0000	0.1111	0.0000	0.0064	0.0547	0.1505	0.2587	0.3568
49	-1.0000	-0.1111	0.0000	0.0058	0.0518	0.1455	0.2530	0.3512
50	1.0000	0.1111	0.0000	0.0053	0.0490	0.1407	0.2474	0.3457

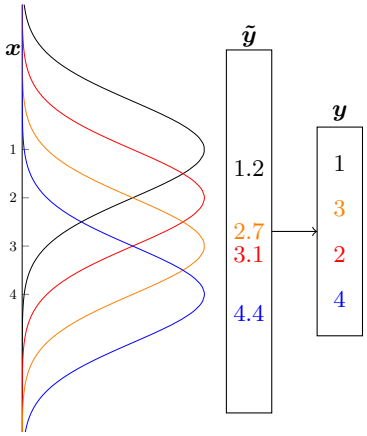

Fig. 1. Inducing error in ranked list $\boldsymbol{x}$ containing four items, using Gaussian Perturbation. Each item is treated as a Normal distribution with mean equal to its rank and constant standard deviation (in this case, $\sigma = 1$). A ranked list with error $\boldsymbol{y}$ is obtained by sampling from the Normal distributions (to obtain the latent values $\tilde{\boldsymbol{y}}$), then ordering by the sample value. We can see that a value of 3.1 was sampled from the Normal distribution with mean 2, and 2.7 was sampled from the distribution with mean 3, pushing item 3 to the 2nd rank and item 2 to the 3rd rank.

Using Gaussian Perturbation, we can compute the expected τ as a function of n and σ. The expected value of τ is given as:

$$\begin{aligned}\mathbb{E}[\tau] &= \mathbb{E}\left[\frac{2\sum_{1\leq i<j\leq n} c_{ij}}{n(n-1)/2} - 1\right] \\ &= \frac{2\sum_{1\leq i<j\leq n} \mathbb{E}\left[c_{ij}\right]}{n(n-1)/2} - 1 \end{aligned} \tag{3}$$

showing that it is dependent on the expected concordance of each pair of items. Let us consider $\boldsymbol{x}$ the ranked list of length n, and the permuted list $\boldsymbol{y}$, based on latent values $\tilde{\boldsymbol{y}}$, which are realisations of Normal random variables $\tilde{\boldsymbol{Y}}$. Given two items i and j, where $x_i < x_j$ in ranked list $\boldsymbol{x}$, then the pair is concordant if $y_i < y_j$ which happens if and only if $\tilde{y}_i < \tilde{y}_j$. Now;

$$\begin{aligned}\mathbb{E}\left[c_{ij}\right] &= 1 \times P(c_{ij} = 1) + 0 \times P(c_{ij} = 0) \\ &= P(\tilde{y}_i < \tilde{y}_j) \\ &= P(\tilde{y}_i - \tilde{y}_j < 0)\end{aligned}$$

and $\tilde{y}_i$ is a sample from a distribution $N(\mu = x_i, \sigma)$. Therefore, $\tilde{y}_i - \tilde{y}_j$ is a sample from a Normal distribution with mean $\mu = x_i - x_j$, and standard deviation $\sqrt{2}\sigma$. After standardising, we obtain:

$$\mathbb{E}\left[c_{ij}\right] = P\left(Z < \frac{x_j - x_i}{\sqrt{2}\sigma}\right) \tag{4}$$

Table 3. The computation time to compute σ when given $\mathbb{E}[\tau]$ and n for Gaussian Perturbation for lists of length 10^1 to 10^6

List Length	10^1	10^2	10^3	10^4	10^5	10^6
Computation Time (sec)	0.002	0.020	0.254	3.083	34.259	379.931

where $Z \sim N(0,1)$ is the standard Normal distribution. Substituting equation 4 back into 3 gives:

$$\mathbb{E}[\tau] = \sum_{1 \le i < j \le n} P\left(Z < \frac{x_j - x_i}{\sqrt{2}\sigma}\right) \frac{4}{n(n-1)} - 1$$

By noticing that $x_j - x_i$ is an integer from 1 to $n-1$, and the sum involves several copies of each such integer, we can simplify the equation further:

$$\mathbb{E}[\tau|n,\sigma] = \sum_{k=1}^{n-1} P\left(Z < \frac{k}{\sqrt{2}\sigma}\right) \frac{4(n-k)}{n(n-1)} - 1 \tag{5}$$

Since $\sigma > 0$ and $n > 1$, each term of the sum in equation 5 is non-negative, showing that $\mathbb{E}[\tau|n,\sigma] \ge 0$. $\mathbb{E}[\tau|n,\sigma]$ can only be zero when all terms of the sum are zero, implying that $\mathbb{E}[\tau|n,\sigma] \to 0$ if and only if $\sigma \to \infty$.

Using equation 5, we can compute the value of σ for a given $\mathbb{E}[\tau]$ and n using a one-dimensional optimisation function. For example, if we want to induce noise so that $\mathbb{E}[\tau] = 0.6$, we find that we must assign $\sigma = 22.46$, 44.48 and 110.54 for $n = 100$, 200 and 500 respectively.

Table 3 provides us with the computation time to perform the one-dimensional optimisation to compute σ. The table shows us that the time to compute σ increases linearly with n. When comparing this to Table 1, we see that Gaussian Perturbation has benefit over the Adjacent Transposition sampling method in terms of the parameter computation time.

3.3 Estimating σ for Gaussian Perturbation

In the previous section, we derived the equation for $\mathbb{E}[\tau]$, dependant on n and σ, and stated that we had to run a one dimensional optimisation over the function to compute σ when given $\mathbb{E}[\tau]$ and n. The computation of σ would be faster if we had a formula that gives the appropriate σ in terms of n and $\mathbb{E}[\tau]$. Unfortunately, σ is embedded in the Normal cumulative density function (CDF) and summed $n-1$ times.

To allow inversion of equation 5 we have made use of the Shah piece-wise approximation [14] to the Standard Normal CDF:

$$P(Z < x) = \begin{cases} x(4.4 - x)/10 + 1/2 & \text{if } x \le 2.2 \\ 0.99 & \text{if } 2.2 < x < 2.6 \\ 1 & \text{if } x \ge 2.6 \end{cases} \tag{6}$$

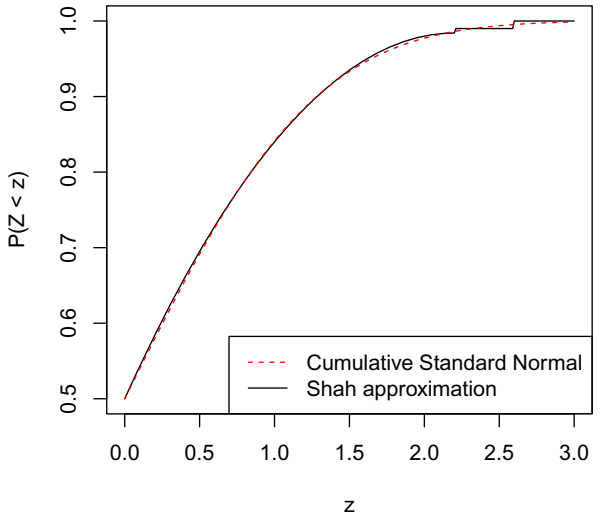

Fig. 2. The Cumulative Standard Normal function and the Shah piece-wise approximation

for $x \geq 0$. The similarity of the Shah approximation to the Standard Normal CDF is shown in Figure 2.

If we assume that $(n-1)/(\sqrt{2}\sigma) \leq 2.2$, we can substitute $P(Z < x)$ with the first piece of the Shah approximation $x(4.4-x)/10 + 1/2$. By making this substitution into equation 5 and simplifying, we obtain the expected τ approximation:

$$\mathbb{E}[\tau|n,\sigma] \approx \frac{4.4\sqrt{2}(n+1)}{30\sigma} - \frac{(n+1)n}{60\sigma^2}$$

which is a quadratic equation in terms of $1/\sigma$. By solving for σ, we obtain:

$$\sigma \approx \frac{n}{4.4\sqrt{2} \pm \sqrt{2 \times 4.4^2 - \frac{60\mathbb{E}[\tau]n}{n+1}}} \tag{7}$$

providing a solution under the condition $\mathbb{E}[\tau] < 4.4^2 \frac{n+1}{n}/30$, meaning that we can compute an approximate σ for all values of n when $0 < \mathbb{E}[\tau] < 0.6453$, and greater values of $\mathbb{E}[\tau]$ as n decreases. Also note that $\sigma \to \infty$ as $\mathbb{E}[\tau] \to 0$, as shown in equation 5.

Of the two solutions, the negative form provides accurate estimates for most of the τ, n combinations, but the positive form provides poor estimates. To examine the estimate's accuracy, we chose a desired value of $\mathbb{E}[\tau]$, used equation 7 to compute the approximate σ, then used equation 5 to compute the obtained $\mathbb{E}[\tau]$.

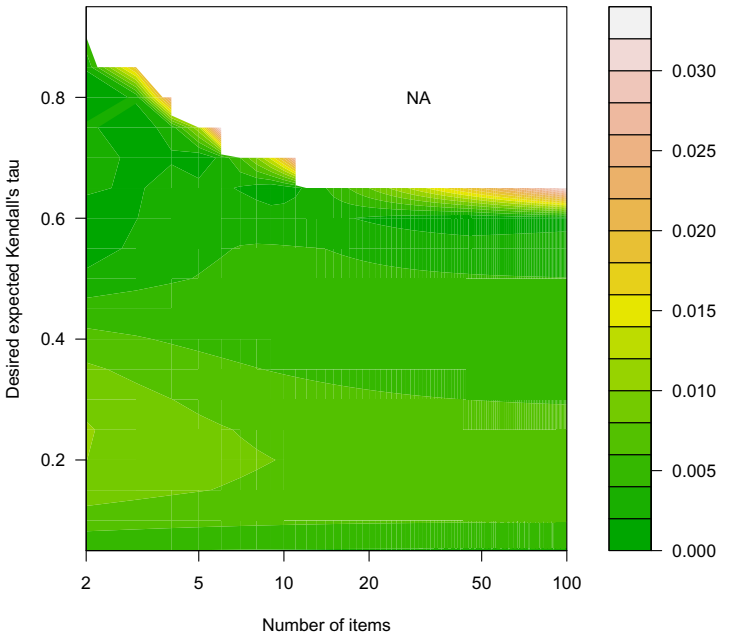

Fig. 3. The absolute difference when comparing the desired expected Kendall's τ, to the obtained expected Kendall's τ, when using the approximate σ computed from equation 7. The difference is computed for varying number if items (n) and desired expected Kendall's τ ($\mathbb{E}[\tau]$).

Figure 3 shows the absolute difference in desired $\mathbb{E}[\tau]$ compared to the obtained $\mathbb{E}[\tau]$, when using an approximate σ. We can see that that the σ estimate is a good estimate since most of the error is smaller than 0.01.

When using equation 7 to compute σ the computation time is less that 10^{-5} seconds for all n.

4 Conclusion

To examine the robustness of systems that take ranked lists as input, we can induce noise by applying a set number of random adjacent transpositions, where the error is measured using Kendall's τ rank correlation. For a fixed list length n, we can ensure a fixed expected τ by keeping the number a random adjacent transpositions constant, allowing a consistent level of noise to be applied to all lists. If the lists are of varying length, it is not clear how many adjacent transposition should be applied to each list to obtain a consistent expected τ over all lists.

In this article, we examined the relationship between the expected τ, the list length n and the number of random adjacent transpositions t. We found that it is possible to compute the expected τ, given n and t, but the computation time rapidly increases with n, making the computation infeasible for $n > 9$.

We also proposed a new method of inducing noise in ranked lists called Gaussian Perturbation, which allows us to derive an equation for the expected τ when given n and its parameter σ. The parameter σ can be computed in reasonable time for large lists (just over six minutes for lists of length 10^6), allowing us to compute σ for various list lengths while keeping the expected τ constant over all lists.

We also provided an approximate solution for σ that requires less that 10^{-5} seconds of computation time for all n.

References

1. Aouf, M., Park, L.A.F.: Approximate document outlier detection using random spectral projection. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 579–590. Springer, Heidelberg (2012)
2. Candès, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Processing Magazine 25(2), 21–30 (2008)
3. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? J. ACM 58(3), 11:1–11:37 (2011)
4. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Computing Surveys (CSUR) 44(1), 1 (2012)
5. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: Proceedings of the 10th International Conference on World Wide Web, pp. 613–622. ACM (2001)
6. Farah, M., Vanderpooten, D.: An outranking approach for rank aggregation in information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, pp. 591–598. ACM, New York (2007)
7. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning. Springer Series in Statistics, vol. 1 (2001)
8. Krüpl, B., Holzinger, W., Darmaputra, Y., Baumgartner, R.: A flight meta-search engine with metamorph. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1069–1070. ACM (2009)
9. Linden, G., Smith, B., York, J.: Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (2003)
10. Liu, Y.T., Liu, T.Y., Qin, T., Ma, Z.M., Li, H.: Supervised rank aggregation. In: Proceedings of the 16th international Conference on World Wide Web, pp. 481–490. ACM (2007)
11. Park, L.A.F.: Fast approximate text document clustering using compressive sampling. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 565–580. Springer, Heidelberg (2011)
12. Ronchetti, E., Field, C., Blanchard, W.: Robust linear model selection by cross-validation. Journal of the American Statistical Association 92(439), 1017–1023 (1997)
13. Schölkopf, B., Smola, A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press (2001)
14. Shah, A.K.: A simpler approximation for areas under the standard normal curve. The American Statistician 39(1), 80–80 (1985)