Exact Paired Permutation Testing Algorithms for NLP Systems

Anonymous ACL submission

Abstract

Significance testing has played a vital role in the development of NLP systems, providing confidence that one system is indeed better than another one. However, many significance tests involve hard computation problems, and so we rely on approximation methods such as Monte Carlo sampling. In this paper, we provide an exact dynamic programming algorithm that runs in quadratic time in the size of the dataset and performs the paired permutation test, a widely used test in comparing two systems, for the case of comparing accuracies between two classification systems. We show that Monte Carlo approximations are often too noisy to reliably determine whether we can reject the null hypothesis. We show that Monte Carlo approximations are often too noisy to reliably determine whether we can reject the null hypothesis with a significance level of $\alpha \approx 0.05$ for any number of sentence N. Additionally, we show that our exact algorithm is more efficient than the approximation algorithm for $N \leq 10K$.

1 Introduction

Statistical hypothesis testing (Lehmann and Romano, 2005) is a fundamental evaluation technique in the sciences. Thus, it should come as no surprise that statistical hypothesis testing is an often employed technique in natural language processing (e.g., Dietterich (1998); Koehn (2004); Ojala and Garriga (2010); Clark et al. (2011); Berg-Kirkpatrick et al. (2012)) for the comparison of competing system, e.g., can we claim that system A has lower error than system B with high confidence.

In this paper, we develop an efficient, exact algorithm for the paired permutation test (Good, 2000), a commonly used method for system comparison in NLP (Yeh, 2000; Dror et al., 2018, 2020; Deutsch et al., 2021). An exact paired permutation test involves a summation over a specific set of permutations. The naïve algorithm to perform this summation runs in exponential time as enumerates all permutations. Thus, practitioners resort to running a Monte Carlo (MC) approximation to the permutation test. However, as with all stochastic simulation, this process introduces additional error when determining whether or not we may reject the null hypothesis. This paper introduces a new dynamic programming algorithm for the performing the computations required by a paired permutation test exactly. Furthermore, we show that the algorithm is efficient when comparing two systems based on accuracy, the standard metric for evaluation in many NLP tasks, e.g. part-of-speech tagging and dependency parsing. Furthermore, in the appendix (App. B), we give an exact quintic algorithm for comparing F_1, but its runtime is too slow to be used on large datasets.

We compare the relative accuracy and efficiency of the exact test and the MC approximation for comparing part-of-speech taggers on the English Universal Dependency Dataset (Nivre et al., 2018). We experimented with the number of MC samples K and the number of sentences N. We find that in all settings, the estimation error of MC can be unreliable even when $\geq 20K$ samples are taken. Additionally, we find that our exact algorithm is faster than using MC with $20K$ and $40K$ samples for datasets with $\leq 6K$ and $\leq 10K$ sentences. Our Python implementation[1] runs the exact permutation test in under three seconds for a dataset with $10K$ sentences. Overall, our exact method appears to be a more practical alternative to MC.

2 Paired Permutation Test

A statistical hypothesis test attempts to reject a null hypothesis $\mathrm{H}_{\varnothing}$ at significance level α. It is common practice to set $\alpha = 0.05$, but there is a movement to lower the scientific standard (Ioannidis, 2018). We now turn to the paired permutation test (Good, 2000), the focus on this work. Suppose we want to

[1] We will release our code publicly upon publication.

evaluate the performance of System A and System B on an input $\mathbf{x}$ with N entries that has a set of true predictions $\mathbf{y}$. Suppose System A predicts $\widehat{\mathbf{a}}$ and System B predicts $\widehat{\mathbf{b}}$. The type of the prediction varies between tasks, e.g., $\mathbf{y} \in \{+1, -1\}^N$ in the simplest case of positive and negative predictions while $\mathbf{y} \in \{1, \ldots, C\}^N$ for general classification problems. In many NLP tasks, such as part-of-speech tagging, dependency parsing, *inter alia*, a sentence-level prediction may be decomposed into word-level predictions. Then $\mathbf{y} \in \{1, \ldots, C\}^{LN}$ where L is the maximum sentence length.

Now, suppose we wish to test the hypothesis that $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$ are different under a scoring function f. We do this by performing a paired permutation test on the null-hypothesis, $\mathrm{H}_\varnothing$, that there is *no* difference between $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$ under f:

$$\mathrm{H}_\varnothing : \ \mathrm{effect}_f\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) \overset{\text{def}}{=} \left| f(\widehat{\mathbf{a}}) - f\left(\widehat{\mathbf{b}}\right) \right| \overset{?}{=} 0 \quad (1)$$

We attempt to reject $\mathrm{H}_\varnothing$ by considering the probability of having seen an effect size as small as observed under the null distribution $P_\varnothing$.

Definition 1. *Given two sets of output, $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$, both of size N, a **paired permutation** (henceforth **permutation**) $\left\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \right\rangle$ is a pair of data that is composed by swapping elements between $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$. Specifically, for entry $n \leq N$, we have that $\widehat{a}'_n = \widehat{a}_n$ and $\widehat{b}'_n = \widehat{b}_n$ (no swap) or $\widehat{a}'_n = \widehat{b}_n$ and $\widehat{b}'_n = \widehat{a}_n$ (swap). The set of all permutations of $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$ is given by $S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$ and $\left| S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \right| = 2^N$.*

Under a paired permutation test, the null-distribution, $P_\varnothing$, is defined to be the uniform distribution over the paired permutations (i.e., $P_\varnothing\left(\left\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \right\rangle\right) = 2^{-N}$). Then, if our observed effect is $o = \mathrm{effect}_f\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right)$ and we have an effect random variable drawn from $P_\varnothing$, $E = \mathrm{effect}_f\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \sim P_\varnothing$, we reject our null-hypothesis if $p = \mathbb{P}(E \leq o) < \alpha$ where α is pre-set. More formally, p can be computed as

$$p = \frac{\sum_{\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \rangle \in S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})} \mathbb{1}\left[\mathrm{effect}_f\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \leq o\right]}{\left| S\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) \right|} \quad (2)$$

The paired permutation test, thus, compares the difference between each individual set of predictions. Peyrard et al. (2021) show that this approach is important for reliable significance testing in NLP.

```
def monte_carlo(â, b̂):
    o ← effect(â, b̂)
    p ← 0
    for i ∈ 1, ..., K:
        ⟨â', b̂'⟩ ← ⟨0, 0⟩
        for n ∈ 1, ..., N:
            ⟨â'_n, b̂'_n⟩ ← RandomSwap(â_n, b̂_n)
        p += (1/K) 𝟙[effect(â', b̂') ≤ o]
    return p
```

Figure 1: Monte Carlo sampling approximation algorithm for the paired permutation test.

Approximating Paired Permutation Test In general, the paired permutation test requires us to compute the sum in (2) The naïve computation clearly runs in exponential time as it requires the enumeration of all $\left| S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \right| = 2^N$ paired permutations. Therefore, most practical implementations of the paired permutation test use a MC approximation, whereby one randomly samples paired permutations to construct an approximate null-distribution. We give this MC algorithm in Fig. 1. Note that the algorithm works for *any* effect function. In the following section, we provide an exact algorithm for the case where effect measures the absolute difference in accuracy.

3 Exact Test for Accuracy

We present a dynamic programming (DP) approach for the paired permutation test. In the general case, the runtime of our DP depends on the chosen effect. However, in the case of accuracy, we are able to derive an efficient exact algorithm.

$$A(\widehat{\mathbf{a}}) \overset{\text{def}}{=} \frac{t(\widehat{\mathbf{a}})}{t(\widehat{\mathbf{a}}) + f(\widehat{\mathbf{a}})} = \frac{t(\widehat{\mathbf{a}})}{N} \quad (3)$$

where $t(\widehat{\mathbf{a}})$ and $f(\widehat{\mathbf{a}})$ are the number of true predictions and false predictions made in $\widehat{\mathbf{a}}$ with regards to $\mathbf{y}$, respectively. We can decompose $t(\widehat{\mathbf{a}})$ among each prediction such that $t(\widehat{\mathbf{a}}) = \sum_{n=1}^{N} t(\widehat{a}_n)$ where $t(\widehat{a}_n) \in \{0, \ldots, L\}$.

The aim of our test is to determine if the p-value

$$p_A = \frac{\sum_{\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \rangle \in S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})} \mathbb{1}\left[\mathrm{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \leq o\right]}{\left| S\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) \right|} \quad (4)$$

is less than the significance level α.

def `perm_test_acc`$(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$:
  $\mathbf{W} \leftarrow \mathbf{0}$
  $\mathbf{W}[0, 0] \leftarrow 1$
  for $n \in 1, \ldots, N$:
    $v_{\text{stay}} \leftarrow t(\widehat{a}'_n) - t\left(\widehat{b}'_n\right)$
    $v_{\text{swap}} \leftarrow t\left(\widehat{b}'_n\right) - t(\widehat{a}'_n)$
    for $v \in \mathbf{W}[n]$:
      $\mathbf{W}[n, v + v_{\text{stay}}]$ **+=** $\frac{1}{2}\mathbf{W}[n-1, v]$
      $\mathbf{W}[n, v + v_{\text{swap}}]$ **+=** $\frac{1}{2}\mathbf{W}[n-1, v]$
  $o \leftarrow \left|t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)\right|$
  $p \leftarrow 0$
  for $v \in \mathbf{W}[N]$:
    if $|v| \leq o$:
      p **+=** $\mathbf{W}[N, v]$
  return p

Figure 2: Dynamic program to find exact p value for the paired-permutation test for accuracy.

Proposition 1. *Given an input* $\mathbf{x}$ *with predictions* $\widehat{\mathbf{a}}$ *and* $\widehat{\mathbf{b}}$ *and true predictions* $\mathbf{y}$*, for any paired permutation* $\left\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \right\rangle$, $\text{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \leq \text{effect}_A\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right)$ *iff* $\left|t(\widehat{\mathbf{a}}') - t\left(\widehat{\mathbf{b}}'\right)\right| \leq \left|t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)\right|$

Proposition 1 indicates that we only need to care about the different in true predictions when performing the paired permutation test for accuracy. We build our DP by constructing a structure $\mathbf{W}$ such that for any $n \in \{0, \ldots, N\}$ and $v \in \mathcal{L}$ where $\mathcal{L} \stackrel{\text{def}}{=} \{-LN, \ldots, LN\}$, $\mathbf{W}[n, l]$ is the probability that a paired permutation $\left\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \right\rangle$ satisfies $t(\widehat{\mathbf{a}}'_{:n}) - t\left(\widehat{\mathbf{b}}'_{:n}\right) = v$ where we define $\widehat{\mathbf{a}}'_{:n}$ to be the first n predictions of $\widehat{\mathbf{a}}'$. Note that we do not consider the absolute value in the DP as we can not decompose an absolute difference into the difference of individual predictions. Once we have $\mathbf{W}$, then the row $\mathbf{W}[N]$ contains the distribution over $t(\widehat{\mathbf{a}}') - t\left(\widehat{\mathbf{b}}'\right)$ and so we can find p_A. The algorithm is formalized as `perm_test_acc` in Fig. 2.[2]

Theorem 1. *Given an input* $\mathbf{x}$ *with predictions* $\widehat{\mathbf{a}}$ *and* $\widehat{\mathbf{b}}$ *and true predictions* $\mathbf{y}$, `perm_test_acc`$(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$ *returns* p_A *in* $\mathcal{O}(LN^2)$ *time and* $\mathcal{O}(LN)$ *space.*

[2] The correctness of `perm_test_acc` is given in Theorem 1. We provide proofs in in App. A.

Metric	Mean	Standard Dev.
Accuracy	0.9543	0.1116
Sentence length	12.08	10.60

Table 1: Distributions for of accuracy and sentence length for POS tagging using Stanza (Qi et al., 2020) on the English UD test dataset (Nivre et al., 2018).

3.1 Practical Implementation

Fig. 2 shows a $\mathbf{W}$ structure that is a $\mathbb{R}^{N \times LN}$ matrix which suggests we need $\mathcal{O}(LN^2)$ space. However, we note that at any iteration n, we only ever need row $\mathbf{W}[n-1]$ and $\mathbf{W}[n]$. Therefore, we only need to maintain two rows of the matrix and so only require $\mathcal{O}(LN)$ space for the algorithm.

4 Experiments

We demonstrate the efficiency of our exact algorithms by simulating paired permutation tests between two systems. In order to have some control over the p-value, N, and L, we randomly generate our two system outputs from a measured distribution. Specifically, we will use the Stanza[3] (Qi et al., 2020) part-of-speech tag statistics when evaluating on the English Universal Dependencies (UD) test set (Nivre et al., 2018). We sample our outputs from the normal distribution where the mean and standard deviation match the rates of Stanza. We further sample the length of each sample sentence according to the distribution of lengths in the test set. The distributions are provided in Tab. 1.

Error Rate of Monte Carlo. We first examine the error rate of the `monte_carlo` as we increase the number of samples used. We sample $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$ using the distributions in Tab. 1, however, we multiply the mean of $\widehat{\mathbf{b}}$ by a factoring in proportion to N. We do this to obtain p-values roughly between 0.001 and 0.1 which is a typical range for α and so a paired permutation test would be required. For each K, we sampled five pairs of systems and run the method five times for each pair, giving 25 data points for each K and N. The reported error rates given in Fig. 3 are the averages of these errors.[4] The mean multiplication factor as well as the average p-value for each N is given in Tab. 2. We see

[3] The code and pre-trained model are both freely accessible at `https://github.com/stanfordnlp/stanza`.

[4] We discard relative errors > 10. We do this to be able to see the noise more clearly in a specific range. However, we note that requiring this additional filter further shows the unreliability of the MC method.

N	**Mean**	**Mean** p	**Standard Dev.** p
250	0.95μ	0.0512	0.1351
500	0.96μ	0.0189	0.0656
1000	0.97μ	0.0153	0.0534
2000	0.98μ	0.0202	0.0536
4000	0.985μ	0.0170	0.0538
8000	0.99μ	0.0360	0.1146

Table 2: Means used for $\widehat{\mathbf{b}}$ distributions in Fig. 3. We use μ to reference the mean 0.9543 given in Tab. 1. The mean and standard deviation of the p-values of the paired permutation test are also given.

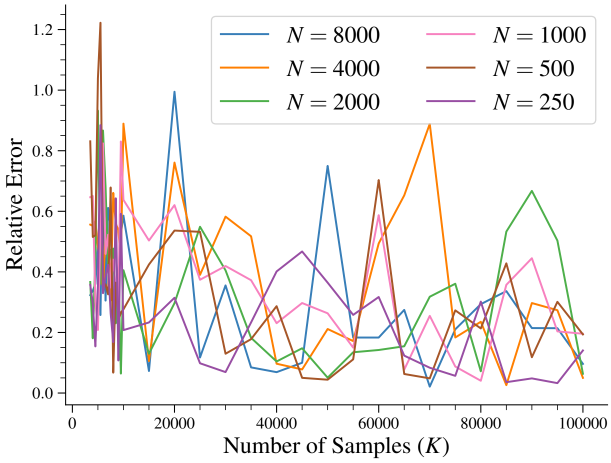

Figure 3: Relative errors of using `monte_carlo` for the paired permutation test. System $\widehat{\mathbf{a}}$ is sampled according to Tab. 1 and system $\widehat{\mathbf{b}}$ is sampled according to Tab. 2.

that the MC approximations are not reliable for all the values of N. While there is a downwards trend as we increase the number of cycles, we observe a lot of noise even when taking 25 attempts per N and K pair. The trend seems most clear until $K = 20{,}000$ at which point we see a lot of noise as K increases. We therefore suggest that 20,000 is the minimum number of samples required when performing a MC paired permutation test, though more is likely better.

Advantages of the Exact Test. When reporting system accuracies in the literature,an exact p-value avoids the estimation error associated with Monte Carlo, as the results above demonstrate. We now show that, empirically, the exact test is *more efficient* than the MC approximation when a large number of samples is taken; this is evinced in Fig. 4. We compare the runtime of `perm_test_acc` against `monte_carlo` for $K = 20{,}000$ and $K = 40{,}000$.[5]

[5]The experiment used an Apple M1 Max processor.

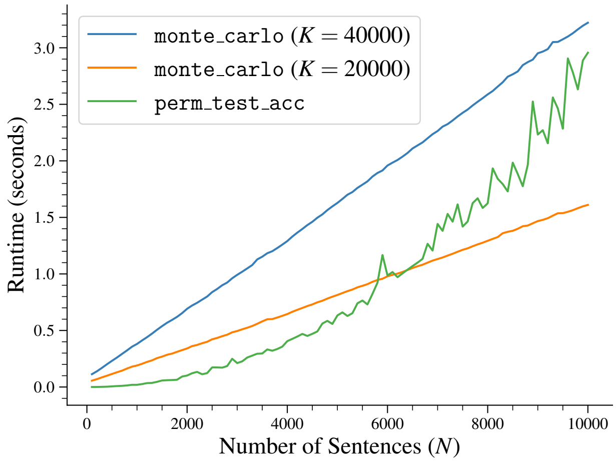

Figure 4: Runtime comparison of `perm_test_acc` and `monte_carlo` as a function of the number of sentences.

We can see that `perm_test_acc` is more efficient than `monte_carlo` with $K = 40{,}000$ and $K = 20{,}000$ for $N < 10{,}000$ and $N < 6{,}000$ (respectively). We note that the average test set size of the UD treebanks[6] is just over $1{,}000$ sentences, and only three treebanks had more than 6,000 sentences.[7] Additionally, the standard split of the commonly used Penn treebank (PTB) (Marcus et al., 1993) provides a test set of about 5,500 sentences. Therefore, the `perm_test_acc` is more efficient than `monte_carlo` for most of the datasets that are used in NLP token-level classification problems.

5 Conclusion

We presented a dynamic programming algorithm to compute the exact p-value of a paired permutation test for the case of difference in accuracy. Our algorithm runs in $\mathcal{O}(LN^2)$ time and requires $\mathcal{O}(LN)$ space. We empirically show that when using MC approximation techniques, we often require $K > LN$ samples to obtain a "good enough" approximation. Therefore, not only is the MC method imprecise, it is also often slower than our exact algorithm for commonly used datasets. We also note that our dynamic program can be extended to compute exact p-values for the paired permutation test using other metrics such as the difference in F_1 scores (see App. B). However, these may by impractical for reasonably sized dataset.

[6]We examined a total of 129 treebanks as some languages have multiple treebanks.

[7]These were Czech, Japanese, and Russian which had test set sizes of roughly 10,000, 8,000, and 6,500 (respectively).

References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 995–1005. ACL.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Daniel Deutsch, Rotem Dror, and Dan Roth. 2021. A statistical analysis of summarization evaluation metrics using resampling methods. *Transactions of the Association for Computational Linguistics*, 9:1132–1146.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Rotem Dror, Lotem Peled-Cohen, Segev Shlomov, and Roi Reichart. 2020. *Statistical Significance Testing for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Phillip Good. 2000. *Permutation Tests A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer.

John P. A. Ioannidis. 2018. The Proposal to Lower *P* Value Thresholds to .005. *JAMA*, 319(14):1429–1430.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Erich Leo Lehmann and Joseph P. Romano. 2005. *Testing Statistical Hypotheses*. Springer.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguistics*, 19(2):313–330.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phuong Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Kyung-Tae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Markus Ojala and Gemma C. Garriga. 2010. Permutation tests for studying classifier performance. *The Journal of Machine Learning Research*, 11:1833–1863.

Maxime Peyrard, Wei Zhao, Steffen Eger, and Robert West. 2021. Better than average: Paired evaluation of NLP systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2301–2315. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the Association for Computational Linguistics: System Demonstrations*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

A Proofs for Section §3 (Exact Test for Accuracy)

Proposition 1. *Given an input* $\mathbf{x}$ *with predictions* $\widehat{\mathbf{a}}$ *and* $\widehat{\mathbf{b}}$ *and true predictions* $\mathbf{y}$*, for any paired permutation* $\left\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \right\rangle$, $\mathrm{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \le \mathrm{effect}_A\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right)$ *iff* $\left|t(\widehat{\mathbf{a}}') - t\left(\widehat{\mathbf{b}}'\right)\right| \le \left|t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)\right|$

Proof.

$$\begin{aligned}
\mathrm{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \le \mathrm{effect}_A\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) &\iff \left|A(\widehat{\mathbf{a}}') - A\left(\widehat{\mathbf{b}}'\right)\right| \le \left|A(\widehat{\mathbf{a}}) - A\left(\widehat{\mathbf{b}}\right)\right| \\
&\iff \left|\frac{t(\widehat{\mathbf{a}}') - t\left(\widehat{\mathbf{b}}'\right)}{|\mathbf{y}|}\right| \le \left|\frac{t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)}{|\mathbf{y}|}\right| \\
&\iff \left|t(\widehat{\mathbf{a}}') - t\left(\widehat{\mathbf{b}}'\right)\right| \le \left|t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)\right|
\end{aligned}$$

■

Theorem 1. *Given an input* $\mathbf{x}$ *with predictions* $\widehat{\mathbf{a}}$ *and* $\widehat{\mathbf{b}}$ *and true predictions* $\mathbf{y}$*,* `perm_test_acc`$(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$ *returns* p_A *in* $\mathcal{O}(LN^2)$ *time and* $\mathcal{O}(LN)$ *space.*

Proof. We first prove that for all $n \in \{0, , \dots, N\}$, $\mathbf{W}[n]$ is the probability distribution $\mathbb{P}\left(\mathrm{effect}_A\left(\widehat{\mathbf{a}}'_{:n}, \widehat{\mathbf{b}}'_{:n}\right)\right)$.
Base case: Then $n = 0$. We have that $\mathbf{W}[0, 0] = 1$ and $\mathbf{W}[0, v] = 0$ for all $v \in \mathcal{L} \setminus \{0\}$.
Inductive step: Assume that $\mathbf{W}[n-1]$ is the probability distribution $\mathbb{P}(\mathrm{effect}_A(\widehat{\mathbf{a}}'_{:(n-1)}, \widehat{\mathbf{b}}'_{:(n-1)})$. Let $v \in \mathcal{L}$ be a candidate difference and $v' = t(\widehat{a}'_n) - t\left(\widehat{b}'_n\right)$. We know that $\left\langle \widehat{a}'_n, \widehat{b}'_n \right\rangle$ is $\left\langle \widehat{a}_n, \widehat{b}_n \right\rangle$ with probability $\frac{1}{2}$ or $\left\langle \widehat{b}_n, \widehat{a}_n \right\rangle$ with probability $\frac{1}{2}$. Therefore, $v' = t(\widehat{a}_n) - t\left(\widehat{b}_n\right)$ with probability $\frac{1}{2}$ or $v' = t\left(\widehat{b}_n\right) - t(\widehat{a}_n)$ with probability $\frac{1}{2}$. Then

$$\mathbb{P}\left(\mathrm{effect}_A\left(\widehat{\mathbf{a}}'_{:n}, \widehat{\mathbf{b}}'_{:n}\right) = v\right) = \frac{1}{2}\mathbb{P}\left(\mathrm{effect}_A\left(\widehat{\mathbf{a}}'_{:(n-1)}, \widehat{\mathbf{b}}'_{:(n-1)}\right) = v - v'\right) = \frac{1}{2}\mathbf{W}[n-1, v-v']$$

This is exactly what is done from Line 7 to Line 9 in Fig. 2. Therefore, $\mathbf{W}[n]$ is the probability distribution $\mathbb{P}(\mathrm{effect}_A\left(\widehat{\mathbf{a}}'_{:n}, \widehat{\mathbf{b}}'_{:n}\right)$.

Line 11 to Line 14 in Fig. 2 construct the p-value using the following equation

$$\begin{aligned}
\sum_{v \in \mathbf{W}[N]} \mathbb{1}[|v| \le o]\mathbf{W}[N, v] &= \sum_{v \in \mathbf{W}[N]} \mathbb{1}[|v| \le o]\mathbb{P}\left(\mathrm{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) = v\right) \\
&= \mathbb{P}\left(\mathrm{effect}_A\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \le \mathrm{effect}_A\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right)\right) = p_A
\end{aligned}$$

where $o = \left|t(\widehat{\mathbf{a}}) - t\left(\widehat{\mathbf{b}}\right)\right|$

The algorithm runs over two nested for-loops of sizes $\mathcal{O}(N)$ and $\mathcal{O}(LN)$ respectively. As the inner loop does constant amount of work per iteration, `perm_test_acc` runs in $\mathcal{O}(LN^2)$ time. The space complexity is discussed in §3.1. ■

B Exact Paired Permutation Test for F_1

We now derive a similar DP algorithm for the case of the F_1 score which we define as

$$F_1(\mathbf{x}) = \frac{t^+(\widehat{\mathbf{a}})}{t^+(\widehat{\mathbf{a}}) + \frac{1}{2}f(\widehat{\mathbf{a}})} \tag{5}$$

where $t^+(\widehat{\mathbf{a}})$ is the number of true positive predictions made in $\widehat{\mathbf{a}}$ with regards to $\mathbf{y}$.

def perm_test_F1$(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}, K)$:
  $\mathbf{W} \leftarrow \mathbf{0}$
  $\mathbf{W}[0, \langle 0,0,0,0\rangle] \leftarrow 1$
  for $n \in 1, \ldots, N$:
    for $\langle t_a^+, f_a, t_b^+, f_b\rangle \in \mathbf{W}[n]$:
      $v_{\text{stay}} \leftarrow \left\langle t_a^+ + t^+(\widehat{a}'_n), f_a + f(\widehat{a}'_n), t_b^+ + t^+\left(\widehat{b}'_n\right), f_b + f\left(\widehat{b}'_n\right)\right\rangle$
      $v_{\text{swap}} \leftarrow \left\langle t_a^+ + t^+\left(\widehat{b}'_n\right), f_a + f\left(\widehat{b}'_n\right), t_b^+ + t^+(\widehat{a}'_n), f_b + f(\widehat{a}'_n)\right\rangle$
      $\mathbf{W}[n, v_{\text{stay}}]$ **+=** $\frac{1}{2}\mathbf{W}[n-1, \langle t_a^+, f_a, t_b^+, f_b\rangle]$
      $\mathbf{W}[n, v_{\text{swap}}]$ **+=** $\frac{1}{2}\mathbf{W}[n-1, \langle t_a^+, f_a, t_b^+, f_b\rangle]$
  $o \leftarrow \left| \frac{t^+(\widehat{\mathbf{a}})}{t^+(\widehat{\mathbf{a}}) + \frac{1}{2}f(\widehat{\mathbf{a}})} - \frac{t^+\left(\widehat{\mathbf{b}}\right)}{t^+\left(\widehat{\mathbf{b}}\right) + \frac{1}{2}f\left(\widehat{\mathbf{b}}\right)} \right|$
  $p \leftarrow 0$
  for $\langle t_a^+, f_a, t_b^+, f_b\rangle \in \mathbf{W}[N]$:
    if $\left| \frac{t_a^+}{t_a^+ + \frac{1}{2}f_a} - \frac{t_b^+}{t_b^+ + \frac{1}{2}f_b} \right| \leq o$:
      p **+=** $\mathbf{W}[N, \langle t_a^+, f_a, t_b^+, f_b\rangle]$
  return p

Figure 5: Dynamic program to find exact p value for the paired-permutation test for F_1.

The aim of our significance test is to decide whether the p-value

$$p_{F_1} = \frac{\sum_{\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\rangle \in S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})} \mathbb{1}\left[\text{effect}_{F_1}\left(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}'\right) \leq o\right]}{\left|S\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right)\right|} \tag{6}$$

is less than the significance level α where $\text{effect}_{F_1}\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) \overset{\text{def}}{=} \left|F_1(\widehat{\mathbf{a}}) - F_1\left(\widehat{\mathbf{b}}\right)\right|$. Unfortunately, unlike accuracy, we cannot decompose the F_1 score into a single additive component. We can write $\text{effect}_{F_1}(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$ as

$$\text{effect}_{F_1}\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) = \left| \frac{t^+(\widehat{\mathbf{a}})}{t^+(\widehat{\mathbf{a}}) + \frac{1}{2}f(\widehat{\mathbf{a}})} - \frac{t^+\left(\widehat{\mathbf{b}}\right)}{t^+\left(\widehat{\mathbf{b}}\right) + \frac{1}{2}f\left(\widehat{\mathbf{b}}\right)} \right|$$

Therefore, we have four variables that we can decompose along the data points, $t^+(\widehat{\mathbf{a}})$, $f(\widehat{\mathbf{a}})$, $t^+\left(\widehat{\mathbf{b}}\right)$, and $f\left(\widehat{\mathbf{b}}\right)$. We construct a similar DP to perm_test_acc, however instead of maintaining the difference in true predictions, we maintain a tuple of the four aforementioned variables. We give this algorithm as perm_test_F1 in Fig. 5 As each variable can be any of $\mathcal{O}(LN)$ values, this makes our DP have a runtime of $\mathcal{O}(L^4N^5)$. Unfortunately, while the algorithm is polynomial in time, the quintic factor makes it impractical for common NLP datasets as described in §4

Theorem 2. *Given an input* $\mathbf{x}$ *with predictions* $\widehat{\mathbf{a}}$ *and* $\widehat{\mathbf{b}}$ *and true predictions* $\mathbf{y}$, perm_test_F1$(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})$ *returns* p_{F_1} *in* $\mathcal{O}(L^4N^5)$ *time and* $\mathcal{O}(L^4N^4)$ *space.*

Proof. For any $n \in \{0, , \ldots, N\}$ and $\langle t_a^+, f_a, t_b^+, f_b\rangle \in \mathcal{L}^4$, we define $E_n\left(t_a^+, f_a, t_b^+, f_b\right)$ to be the event that $t(\widehat{\mathbf{a}}'_{:n}) = t_a^+$, $f(\widehat{\mathbf{a}}'_{:n}) = f_a$, $t\left(\widehat{\mathbf{b}}'_{:n}\right) = t_b^+$, and $f\left(\widehat{\mathbf{b}}'_{:n}\right) = f_b$. We first prove that $\mathbf{W}[n]$ is the probability distribution over the tuples such that

$$\mathbf{W}[n, \langle t_a^+, f_a, t_b^+, f_b\rangle] = \mathbb{P}\left(E_n(t_a^+, f_a, t_b^+, f_b)\right) \tag{7}$$

Base case: Then $n = 0$. We have that $\mathbf{W}\left[0, \langle 0,0,0,0\rangle\right] = 1$ and $\mathbf{W}[0, v] = 0$ for all $v \in \mathcal{L} \setminus \{\langle 0,0,0,0\rangle\}$.

Inductive step: Assume that $\mathbf{W}[n-1]$ is the probability distribution described in (7). We know that $\left\langle \widehat{a}'_n, \widehat{b}'_n \right\rangle$ is $\left\langle \widehat{a}_n, \widehat{b}_n \right\rangle$ with probability $\frac{1}{2}$ or $\left\langle \widehat{b}_n, \widehat{a}_n \right\rangle$ with probability $\frac{1}{2}$. Then, if we let $\langle t_a^+, f_a, t_b^+, f_b \rangle \in \mathcal{L}^4$, we can find the following probability

$$\begin{aligned}\mathbb{P}\left(E_n(t_a^+, f_a, t_b^+, f_b)\right) &= \frac{1}{2}\mathbb{P}\left(E_{n-1}\left(t_a^+ - t(\widehat{a}'_n), f_a - f(\widehat{a}'_n), t_b^+ - t\left(\widehat{b}'_n\right), f_b - f\left(\widehat{b}'_n\right)\right)\right) \\ &= \frac{1}{2}\mathbf{W}\left[n-1, \left\langle t_a^+ - t(\widehat{a}'_n), f_a - f(\widehat{a}'_n), t_b^+ - t\left(\widehat{b}'_n\right), f_b - f\left(\widehat{b}'_n\right)\right\rangle\right]\end{aligned}$$

This is exactly what is done from Line 8 to Line 9 in Fig. 5. Therefore, $\mathbf{W}[n]$ is the probability distribution $\mathbb{P}\left(E_n(t_a^+, f_a, t_b^+, f_b)\right)$.

Line 11 to Line 14 in Fig. 5 construct the p-value using the following equation

$$\sum_{\langle t_a^+, f_a, t_b^+, f_b \rangle \in \mathbf{W}[N]} \mathbb{1}\left[\left|\frac{t_a^+}{t_a^+ + \frac{1}{2}f_a} - \frac{t_b^+}{t_b^+ + \frac{1}{2}f_b}\right| \leq o\right]\mathbf{W}[N, \langle t_a^+, f_a, t_b^+, f_b \rangle] \tag{8}$$

$$= \sum_{\langle t_a^+, f_a, t_b^+, f_b \rangle \in \mathbf{W}[N]} \mathbb{1}\left[\left|\frac{t_a^+}{t_a^+ + \frac{1}{2}f_a} - \frac{t_b^+}{t_b^+ + \frac{1}{2}f_b}\right| \leq o\right]\mathbb{P}\left(E_N(t_a^+, f_a, t_b^+, f_b)\right) \tag{9}$$

$$= \sum_{\langle \widehat{\mathbf{a}}', \widehat{\mathbf{b}}' \rangle \in S(\widehat{\mathbf{a}}, \widehat{\mathbf{b}})} \mathbb{1}\left[\text{effect}_{F_1}(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}') \leq o\right]\mathbb{P}\left(E_N\left(t(\widehat{\mathbf{a}}'), f(\widehat{\mathbf{a}}'), t\left(\widehat{\mathbf{b}}'\right), f\left(\widehat{\mathbf{b}}'\right)\right)\right) \tag{10}$$

$$= \mathbb{P}\left(\text{effect}_{F_1}(\widehat{\mathbf{a}}', \widehat{\mathbf{b}}') \leq o\right) = p_{F_1} \tag{11}$$

The algorithm runs over two nested for-loops of sizes $\mathcal{O}(N)$ and $\mathcal{O}(L^4N^4)$ respectively. As the inner loop does constant amount of work per iteration, `perm_test_F1` runs in $\mathcal{O}(L^4N^5)$ time. We need to store two rows of $\mathbf{W}$ at any given time. Therefore the space complexity is $\mathcal{O}(L^4N^4)$ ■