Zhao Chen Mason duplicate-detection-ACM

Conference Paper · December 2021

DOI: 10.1145/3486622.3493928

3 authors:

Yanchang Zhao
The Commonwealth Scientific and Industrial Research Organisation
62 PUBLICATIONS **1,320** CITATIONS

SEE PROFILE

Haohui Chen
The Commonwealth Scientific and Industrial Research Organisation
32 PUBLICATIONS **1,156** CITATIONS

SEE PROFILE

Claire M. Mason
The Commonwealth Scientific and Industrial Research Organisation
66 PUBLICATIONS **1,706** CITATIONS

SEE PROFILE

A Framework for Duplicate Detection from Online Job Postings

Yanchang Zhao
Data61, CSIRO
Australia
yanchang.zhao@csiro.au

Haohui Chen
Data61, CSIRO
Australia
caronhaohui.chen@data61.csiro.au

Claire M. Mason
Data61, CSIRO
Australia
claire.mason@data61.csiro.au

ABSTRACT

Online job boards have greatly improved the efficiency of job searching and have also provided valuable data for labour market research. However, there are a high proportion of duplicate job postings in most (if not all) job boards, because recruiters and job boards seek to improve their coverage of the market by integrating job postings from many different sources. These duplicate postings undermine the usability of job boards and the quality of labour market analytics derived from them. In this paper, we tackle the challenging problem of duplicate detection from online job postings. Specifically, we design a framework for duplicate detection from online job postings and, under the framework, implement and test 24 methods built with four different tokenisers, three vectorisers and six similarity measures. We conduct a comparative study and experimental evaluation of the 24 methods and compare their performance with a baseline approach. All methods are tested with a real-world dataset from a job boarding platform and are evaluated with six performance metrics. The experiment reveals that the top two methods are *Overlap with skip-gram (OS)* and *Overlap with n-gram (OG)*, followed by *TFIDF-cosine with n-gram (TCG)* and *TFIDF-cosine with skip-gram (TCS)*, and that all above four methods outperform the baseline approach in detecting duplicates.

CCS CONCEPTS

• **Applied computing** → **Document analysis**; • **Computing methodologies** → *Information extraction*; • **Information systems** → Data cleaning.

KEYWORDS

duplicate detection, job posting, text mining, document analysis

ACM Reference Format:
Yanchang Zhao, Haohui Chen, and Claire M. Mason. 2021. A Framework for Duplicate Detection from Online Job Postings. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3486622.3493928

1 INTRODUCTION

There are many online job/career platforms, such as SEEK, CareerOne, Adzuna and LinkedIn, which have collected a large volume of job postings. However, due to the fact that recruiters often post job postings to multiple platforms and that platform providers scrape job postings from one another (to improve their coverage of the market), duplicate job postings are common. A recent study by Jijkoun [9] shows that an average job ad can be re-posted two to five times and that the fraction of duplicates can be as high as 50-80%. Identifying duplicate job postings is important for two reasons. First, the usability of the platform is diminished when search results include many duplicates or users receive multiple alerts for the same job. Second, job postings have now become a valued source of data for monitoring labour markets since they provide near-to-real-time data on employer demand for workers and skills [5, 8, 14]. The efficiency of users' search for relevant job postings and the analysis of job trend and skills in demand is substantially impacted when duplicate job postings are so common.

In this paper we describe the development of a framework for duplicate detection from online job postings. Under the framework, we have developed and experimentally evaluated 24 methods on a real-world dataset captured from Adzuna Australia, a job board and search engine for the Australian job market. The 24 methods are built with

- four different tokenisers: word tokeniser, word tokeniser with stop words removal, n-gram tokeniser and skip-gram tokeniser, which are referred to respectively as *word*, *word-2*, *n-gram* and *skip-gram* in the rest of this paper,
- three vectorisers: Document-Term Matrix (DTM), Term Frequency - Inverse Document Frequency (TFIDF) and Latent Semantic Analysis (LSA) [6], and
- six similarity measures: *Jaccard*, *Overlap*, *Cosine*, *LSA-cosine*, *TFIDF-Euclidean* and *TFIDF-cosine*.

In our experiments, the above methods were compared with Apollo [2], an existing approach for duplicate detection from job postings. The performance of each method was evaluated with six measures: correlation, AUC, accuracy, precision, recall and F1-score. Experimental results show that *Overlap with skip-gram (OS)* and *Overlap with n-gram (OG)* were the best two methods, followed by *TFIDF-cosine with n-gram (TCG)* and *TFIDF-cosine with skip-gram (TCS)*, and that all of the above four methods outperformed Apollo [2].

Contributions of this work include:

- a framework for detecting duplicates from job postings,
- a blocking and re-grouping method to speed up the process,
- a comparative study of 24 methods built with four different tokenisers, three vectorisers and six similarity measures, and

WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia

ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493928

- an experimental evaluation of the above methods and comparison with an existing approach on real-world job posting data.

2 RELATED WORK

2.1 Near-Duplicate Web Page Detection

Broder et al. [1] developed a method to determine the syntactic similarity of documents and used it to cluster documents on the Internet, with possible applications like a "lost-and-found" service for documents, filtering web search results and identifying plagiarism. Lecocq [11] summarised multiple methods for near-duplicate content detection, including bag of words, shingling, hashing, MinHash and SimHash. Henzinger [7] performed an evaluation of two near-duplicate detection algorithms on a set of 1.6B web pages, and the results showed that both algorithms achieved high precision for finding near-duplicate pairs on different sites, but worked poorly on pairs from the same site. Manku et al. [12] investigated the problem of detecting near-duplicates for web crawling, i.e., to assess whether a newly crawled web page is a near-duplicate of a previously crawled one. Their study showed that Charikar's SimHash and fingerprinting technique [4] is practically useful for identifying near-duplicate web pages. They also developed a method to quickly find all fingerprints that are different from a given one, which is fast in both online and batch near-duplicate detection.

2.2 Detecting Duplicate Postings in Online Classifieds

Kaggle [10] hosted a competition on detecting duplicate postings posted in online classifieds in 2016, where duplicate postings were to be identified based on their contents including Russian language text and images. The postings were collected from online marketplaces where people buy or sell items, rather than job postings. The competition provided a training dataset, which contained pairs of postings that were duplicate and not duplicates. In our work, we take an unsupervised approach, which makes our approach better in that in many instances training datasets are unavailable or very costly to obtain.

2.3 Duplicate Job Ads Detection

Jijkoun [9] presented a methodology used at Textkernel for duplicate detection from online job postings, Their study suggested that the fraction of duplicates can be as high as 50-80% in job postings crawled from online job boards and other sites. In that method, a statistical classifier was trained to predict whether two postings referred to the same job. Shingling was used to measure the textual similarity between two documents and a locality sensitive hashing scheme and ElasticSearch were used for improving efficiency. Burk et al. [2] developed a system named Apollo for near-duplicate job postings detection in the online recruitment domain. With a range of techniques including blocking, shingling, boilerplate text removal and Jaccard similarity, their experimental results showed that the system achieved higher precision, recall and F-score than SimHash and Shingling. In their method, the time gap between job postings was not considered, fixed heuristic thresholds and 5-shingle (i.e., 5-gram) were used and stop words were not removed. We take this method as a baseline and compare our methods with it in experiments.

3 PROBLEM STATEMENT

There are various types of duplicate job postings. For example, a job ad could have been posted on multiple sites and therefore end up with multiple copies when crawling. Even on a single site, a job ad can be re-posted multiple times within a few weeks and sometimes even within a few days. From the collected data, we found some job ad series, composed of a single job ad being re-posted multiple times within a short timeframe. Some of them are re-posted after two months, which are likely to be re-advertising when a vacancy is not filled yet. However, there are often also a series of postings with the same ad re-posted almost daily or weekly, which are likely to be spam or posting by bots. Based on exploration of the job postings data and discussion with domain experts, we applied a window of 60 days to identify duplicates from series. That is, a re-posting of an original ad after 60 days is taken as a new job ad, rather than a duplicate. The same window size has been used for job ad de-duplication in existing work [3].

We define a duplicate job posting as follows. A job posting is a duplicate if it has both the same title and the same location and has very similar job description to one or more other postings posted within 60 days prior to it. Assume A and B are two job postings. A is a duplicate of B if

$$\begin{cases} \text{Timestamp}(A) > \text{Timestamp}(B), \\ 0 \leq \text{Date}(A) - \text{Date}(B) \leq 60, \\ \text{Title}(A) = \text{Title}(B), \\ \text{Location}(A) = \text{Location}(B), \text{and} \\ \text{Sim}(A, B) \geq \tau, \end{cases} \tag{1}$$

where Date, Timestamp, Title and Location are respectively the post date, timestamp, job title and location of a job posting, Sim is a similarity measure and τ is a similarity threshold.

If two postings have very similar job descriptions but different locations or different job titles, they are not considered duplicate of each other, because most likely they are two vacancies at different locations or with similar but different roles.

4 METHODOLOGY

This section presents our methodology for duplicate detection. A framework of it is shown in Figure 1, which is composed of four steps. First, the body text of job postings is cleaned by converting to lower case, removing non-alphanumeric symbols, collapsing multiple spaces and removing stop words. Next, tokens are extracted with four tokenisers and job postings are vectorised with Document-Term Matrix (DTM), Term Frequency - Inverse Document Frequency (TFIDF) and Latent Semantic Analysis (LSA) [6]. After that, six similarity measures are used to calculate the similarity score between every pair of job postings. Finally, those pairs with similarity scores above a given threshold are identified as duplicates. Under the framework, 24 methods with the four different tokenisers and the six similarity measures are developed.

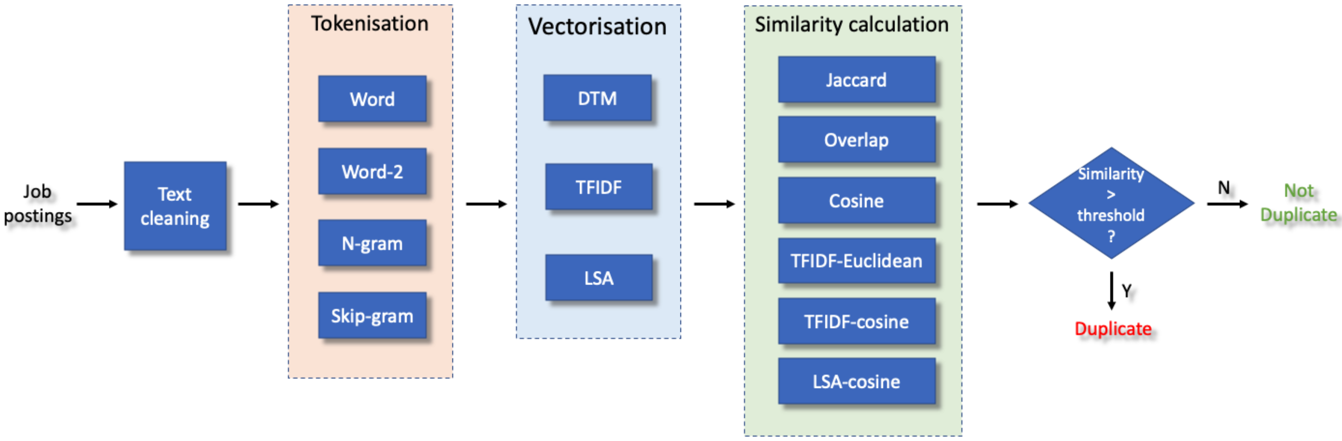

Figure 1: A Framework for Duplicate Detection from Job Postings

4.1 Tokenisers

We use four different tokenisers: word tokeniser, word tokeniser with stop words removal (referred to as "word-2"), n-gram tokeniser and skip-gram tokeniser.

4.1.1 Word Tokeniser. A word tokeniser simply takes every single word as a token by splitting text with spaces and punctuation marks. Here we use a short sentence "*This is a simple example of text tokenisation*" as an example. The tokens produced by word tokeniser are shown in the second row in Table 1. If stop words are removed, the tokens left are shown in the third row ("word-2"), where stop words, such as "this", "is", "a" and "of", are excluded.

4.1.2 N-Gram. Rather than taking every single word as a token, n-gram extracts n consecutive words as tokens. This can be done with or without stop words removal. Since stop words contribute little to analysis in this work, they are removed before extracting n-grams. The 2-grams for the same sentence are shown in the fourth row ("n-gram (n=2)") of Table 1.

4.1.3 Skip-Gram. Skip-grams are similar to n-grams, but the former skips over gaps when extracting tokens. More specifically, a k-skip-n-gram is a subsequence of n words where the words occur no more than k words away from each other. Skip-gram overcomes the data sparsity problem with n-gram. The 1-skip-2-grams with stop word removal produced from the same text are shown as the last row of Table 1, which includes "simple text" and "example tokenisation", in addition to all 2-grams.

4.2 Vectorisation

With tokens extracted from job postings, a document-term matrix and a TFIDF matrix are built to vectorise job posting data. In the matrices, each row represents a document (i.e., a job posting) and each column represents a term (i.e., a token). In a document-term matrix M, an entry $m_{i,j}$ is the frequency of term j in document i. In a TFIDF matrix W, an entry is defined as $w_{i,j} = \mathrm{tf}_{i,j} \times \log(n/\mathrm{df}_j)$, where $\mathrm{tf}_{i,j}$ is the frequency of term j in document i, df_j is the number of documents containing term j and n is the total number of documents. In addition to DTM and TFIDF, we also use Latent Semantic Analysis [6] for vectorisation. The vectorisation produced by the above three methods is then used to calculate the similarity between every pair of postings.

Table 1: An Example Illustrating the Four Tokenisers

Tokeniser	Tokens
word	this, is, a, simple, example, of, text, tokenisation
word-2	simple, example, text, tokenisation
n-gram (n=2)	simple example, example text, text tokenisation
skip-gram (n=2, k=1)	simple example, simple text, example text, example tokenisation, text tokenisation

4.3 Similarity Measures

Following the above vectorisation, six metrics, Jaccard, Overlap, Cosine, LSA-cosine, TFIDF-Euclidean and TFIDF-cosine, are used to calculate similarity scores between job postings.

4.3.1 Jaccard Index. Assume that $\mathbb{A}$ and $\mathbb{B}$ are respectively token sets of two job postings A and B. The Jaccard index is calculated as

$$\mathrm{Jaccard}(A,B) = \frac{|\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A} \cup \mathbb{B}|}, \tag{2}$$

where "$|\cdot|$" denotes the size of a set. The Jaccard index lies in $[0, 1]$. A value close to one indicates that two postings are very similar to each other, and therefore, are likely to be duplicate.

4.3.2 Overlap. The Overlap index is similar to Jaccard, but the union of two sets in the denominator is replaced with the smaller set. The Overlap index is defined as

$$\mathrm{Overlap}(A,B) = \frac{|\mathbb{A} \cap \mathbb{B}|}{\min(|\mathbb{A}|, |\mathbb{B}|)}, \tag{3}$$

where $\mathbb{A}$ and $\mathbb{B}$ are token sets of two job postings, A and B.

In the context of job postings, sometimes two job postings can have the same job description and therefore are of the same position, but one of them may have some extra information about the recruitment agency or the employer at very beginning, or an extra statement about equal employment opportunity at the very end. In those situations, Overlap would be more effective than Jaccard index in identifying duplicates, which is evidenced by our experimental results in Section 5.4.

4.3.3 Cosine. Cosine similarity is calculated as

$$\text{Cosine}(A, B) = \frac{\mathbf{A} \bullet \mathbf{B}}{||\mathbf{A}|| \ ||\mathbf{B}||}, \tag{4}$$

where **A** and **B** are respectively numeric vectors representing two job postings A and B, "$\bullet$" is dot product and "$|| \cdot ||$" is L_2-norm. For example, **A** and **B** can be the vectors for corresponding job postings from a term-document matrix of term frequency.

4.3.4 TFIDF-Euclidean and TFIDF-cosine. From a term-document matrix, we derive a TFIDF matrix, based on which Euclidean and cosine similarities are calculated.

4.3.5 LSA-cosine. Another method that we use is LSA-cosine, where the Latent Semantic Analysis (LSA) [6] is applied to the TFIDF matrix and then the vectors from the latent space are used to calculate a cosine similarity.

4.4 Speeding Up with Sliding Window, Blocking and Re-Grouping

It is computing intensive to calculate similarity between every single pair of job postings and also consumes RAM when vectorising a large number of postings. To speed up the process of duplicate detection, we use sliding window, blocking and re-grouping techniques.

With the framework given in Figure 1, every single pair of job postings has to be compared to detect all duplicates. This is computationally intensive, with a time complexity of $O(n^2t)$, where n is the number of job postings and t is the time for computing the similarity between two postings. Fortunately, according to the definition of duplicate in Section 3, we do not have to calculate the similarity between every pair of postings. Instead, we need to check only the posting pairs that are within 60 days from each other, and therefore, we use a sliding window of 60 days to reduce the number of similarity calculations.

Moreover, since two duplicate postings have to be of the same job title and same location, we use a blocking technique to speed it up further, by grouping postings by both title and location and then detecting duplicates within every single group in parallel. Nevertheless, the space of job title and location can be very sparse, especially for tiny-sized occupations in a small suburb or in a remote region. This creates many tiny groups, with each consisting of a few job postings only, which brings computing overhead when processing those groups. To reduce the overhead caused by tiny groups, we re-group them by merging very small groups (specifically, with fewer than k job postings) into bigger ones (with at least k job postings) to reduce the number of groups and further improve the processing speed. To find out the optimal group size k, we conduct experiments with various settings of k, ranging from 10 to 2000 (on a computer with configuration details given in Section 5). The experimental results are shown in Figure 2. Note that a group size of zero means no re-regrouping was applied. The left-most two bars ($k = 0$ and $k = 10$) in the figure show that, compared to simple blocking without re-grouping ($k = 0$), the run time is almost halved by re-grouping with group size of 10. The run time is further reduced when increasing group size to 20, 40, and so on. When the group size becomes larger than 300, the run time starts to increase, because, with more job postings within every single group, the RAM needed for processing each group increases and the number of similarity calculations also increases. Based on the above result, we use a group size of 300 to re-group tiny groups in the rest of our experiments. Note that this optimal group size is to a large degree dependent on the average size of job postings, the number of parallel processes and the available RAM of a computer. However, the above method is generic and can be applied to find the optimal k for other applications and computer configurations.

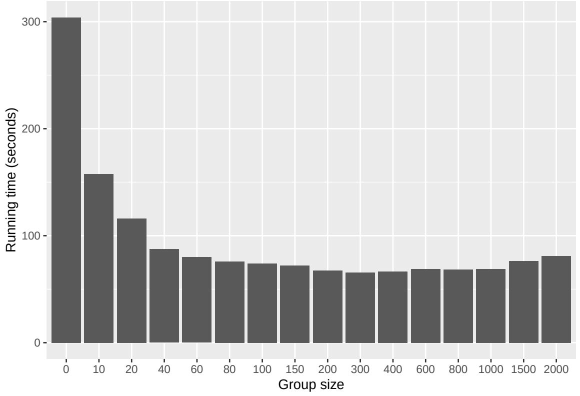

Figure 2: Running Time vs Group Size

4.5 The Process of Duplicate Detection

We process the job postings in a temporal order. Given a set of job postings, the process of detecting duplicates from them is composed of six steps as below.

1) **Loading data.** Load the given dataset of postings (such as those posted in a certain month, week or day) and also the postings within 60 days prior to them.
2) **Removing invalid postings.** Job postings with empty descriptions or invalid post dates are removed.
3) **Blocking.** Split data into groups by job location-title, with each group containing all job postings for a unique combination of location and title.
4) **Removing singles.** Remove every group consisting of one single posting only, because job postings in those groups are not duplicates.
5) **Re-grouping.** Reduce the number of groups by merging small groups (with size $< k$) into bigger ones, with every new group having k or more postings. This reduces the number of processes in parallel computing. The optimal value of k is set according to the experimental results given in Section 4.4 and in Figure 2.

6) **Detecting duplicates within every group.** Follow the process described in Figure 1 to detect duplicates within every group in parallel. For each group, which can consist of a set of multiple combinations of location-title (due to re-grouping):
   a. clean job descriptions by converting them to lower case, removing non-alphanumeric symbols and collapsing multiple white spaces,
   b. extract tokens, such as words, n-grams and skip-grams, from job descriptions,
   c. vectorise the tokens with Document-Term Matrix, TFIDF or LSA,
   d. calculate similarities between each pair of postings, such as Jaccard, Overlap, Cosine, TFIDF-Euclidean, TFIDF-cosine and LSA-cosine,
   e. label a posting as duplicate if there are one or more other postings
      - posted within 60 days prior to it,
      - of the same title and location as it, and
      - having a similarity score greater than a given threshold τ (see Table 4 for values).

Step 6 runs in parallel, with each process handling a group at a time. The number of cores is set based on the number of available CPU cores, the size of RAM and the memory needed by each process. Steps 4 and 5 reduce the number of groups and make it run faster for parallel computing.

5 EXPERIMENTAL EVALUATION

We conducted extensive experiments to evaluate the performance of all the developed methods on a real-world dataset of job postings and compared them with a baseline approach, Apollo [2]. We also compared the impact of different hyper parameters for n-gram and skip-gram, as well as vocabulary size and run time. The experiments were conducted on a MacBook Pro with a 2.9GHz Quad-Core Intel i7 CPU and 16 GB RAM, running MacOS Catalina.

5.1 Data

The job postings data used in our experiments were kindly provided by Adzuna Australia[1], a job board and search engine that has been operating in Australia since 2013. The job postings in their database are not just sourced from job posters who list job postings directly on the Adzuna Australia platform. Job postings that are posted on other Australian online job boards can be listed on the Adzuna Australia platform free of charge. Adzuna Australia also provides online listing of all the job postings listed in one of Australia's largest newspapers. Finally, they scrape data from other websites (e.g., large employers' websites) to capture a wider set of job postings. The range of sources captured in the job postings database means that they are likely to provide good coverage of job postings nationally, but it also increases the potential to capture duplicate job postings.

The dataset that we used for this work represents approximately 9.3 million job postings of 17GB in size, covering the period from March 2015 to May 2019. From the above raw data, we created pairs of job postings, first specifying that each pair of job postings must have the same title and location. These criteria were used to identify possible duplicate job postings that were being treated by the system as two different legitimate job postings. We randomly selected a sample of such pairs and had them labelled as duplicates (dup=1) or non-duplicates (dup=0) by five domain experts. This provided us with a gold dataset of 1498 pairs of job postings. The median length of these postings was 274 words and the maximum length was 1034. The characteristics of these duplicate and non-duplicate pairs are given in Table 2, where the last three columns give length difference of a pair in the number of words. It shows that duplicate pairs tend to have smaller discrepancy in length than non-duplicate ones.

Table 2: Characteristics of Duplicate and Non-duplicate Job Ads

Duplicate Flag	Number of Pairs	Percentage	Length Difference*		
			Min	*Median*	*Max*
dup=0	936	62%	0	90	684
dup=1	562	38%	0	15	326
Total	1498	100%	0	52	684

* Difference in length of two job postings (in number of words)

Table 3: Tokenisers and Vocabulary Sizes

Tokeniser	Settings	Tokens	Stopword Removal	Vocabulary Size
word	-	1-grams	no	13,788
word-2	-	1-grams	yes	13,670
n-gram	n=1,2,3	1-, 2- & 3-grams	yes	374,392
skip-gram	n=1,2; k=1	1- & 2-grams with 1-skip	yes	308,175
Apollo	n=5	5-grams	no	366,183

Table 4: Optimal Thresholds Chosen Using Youden's Index [13, 15]

Tokeniser	Jacc.	Overlap	Cosine	LSA Cosine	TFIDF Euc.	TFIDF Cosine
word	0.6625	0.8741	0.8575	0.9201	0.8498	0.7687
word-2	0.6364	0.8318	0.7654	0.8887	0.7827	0.7581
n-gram	0.5318	0.8053	0.7474	0.9388	0.7742	0.6866
skip-gram	0.5366	0.8061	0.7491	0.9301	0.7772	0.6936

5.2 Experiment Settings

All the methods, together with the Apollo approach [2], were applied to the dataset, and the produced duplicate scores (i.e., similarity scores) were then compared against the labels provided by the five domain experts. The parameter settings of every tokensier and their vocabulary sizes are given in Table 3.

After computing all similarity scores, Youden's Index [13, 15] was used to select the best cut-point for every method (selected thresholds are displayed in Table 4). The threshold for Apollo was set to 0.5, the same setting as that used by Burk et al. [2].

[1] http://www.adzuna.com.au

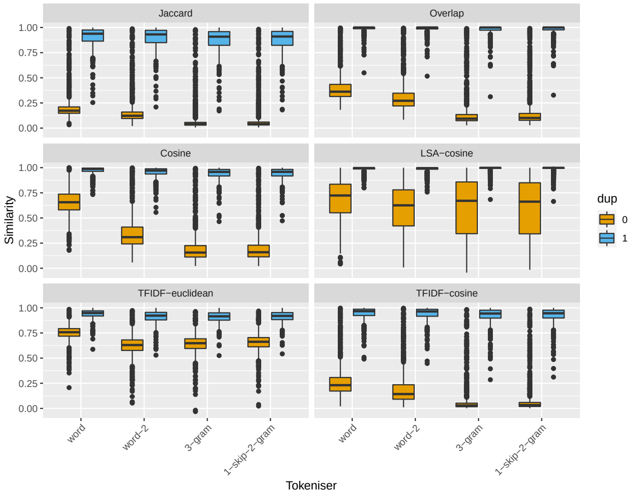

Figure 3: Box-Plots of Similarity Scores for Duplicate (in blue) and Non-Duplicate (in orange) Pairs with Four Tokenisers (see x-axis)

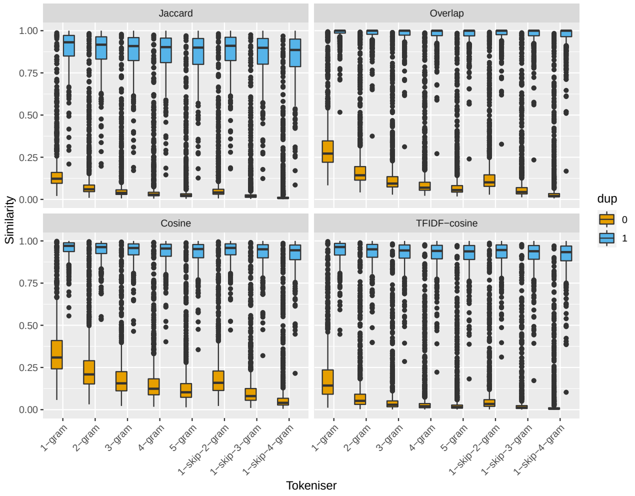

Figure 4: Box-Plots of Similarity Scores for Duplicate Pairs (in blue) and Non-Duplicate Ones (in orange) with Various n-Gram and Skip-Gram Tokenisers (see x-axis)

5.3 Evaluation Measures

To evaluate the effectiveness of duplicate detection, six performance measures are used, namely, correlation (i.e., the correlation between predicted values and actual labels), accuracy ($A = \frac{TP+TN}{TP+FP+TN+FN}$), precision ($P = \frac{TP}{TP+FP}$), recall ($R = \frac{TP}{TP+FN}$), AUC (area under the ROC curve) and F1-score ($F1 = \frac{2 \times P \times R}{P+R}$). In the above formulae, TP and FP are respectively the numbers of true positives and false positives, and TN and FN the numbers of true and false negatives.

5.4 Experimental Results

Figure 3 shows the boxplots of similarity scores produced using various methods. The orange boxes (dup=0) show the distribution of similarity scores between pairs of job postings that are not duplicate, and the blue ones (dup=1) are for duplicate pairs. Within each boxplot, the bar in the middle shows the median of similarity scores and the box shows the interquartile range (IQR). The figure clearly shows that duplicate pairs have similarity scores close to 1, while the similarities between non-duplicate pairs are much lower. The orange boxes in sub-figures for Jaccard, Overlap and TFIDF-cosine are further away from their corresponding blue boxes, which suggests that the three methods are more effective in detecting duplicates than the rest. Out of the above three methods, the top four variations showing best differentiation between duplicates and non-duplicates are *Overlap with n-gram (OG)* and *Overlap with skip-gram (OS)* (see the top-right chart), and *TFIDF-cosine with n-gram (TCG)* and *TFIDF-cosine with skip-gram (TCS)* (see the bottom-right chart), in that their orange and blue boxes are more compact and farther apart from each other.

These descriptive statistics are also confirmed by the experimental results reported in Table 5, where the methods are ordered descendingly first by AUC and then by F1-score. The top four methods for each metric are highlighted in bold. Based on AUC and F1-score, OS and OG are the best methods, followed by TCG and TCS. All the above four methods have better performance than Apollo [2], which uses a 5-gram tokeniser with the Jaccard similarity. Although duplicate pairs tend to have smaller discrepancy in length (see Table 2), there are many duplicate pairs whose lengths are very different from each other, due to extra text about employer or equal employment opportunity added to the very beginning or end of one job posting but not to the other. From checking the gold label dataset, we discovered that extra text was sometimes introduced at the end of a job ad due to recruiters (or job boards) inserting standard wording or contact information at the end of their job postings. This feature of job postings explains why Overlap is more effective than Jaccard for detecting duplicate job postings.

5.5 Results for N-Gram and Skip-Gram

Since TFIDF-Euclidean and LAS-cosine were not as effective as the other methods, they were excluded from further analysis. The remaining four methods were then tested with various parameter settings of n-gram and skip-gram, with results shown in Figure 4 and Table 6. The best performance is achieved by Overlap with 1-skip-4-gram tokeniser. The results also show that, from 1-gram to 5-gram, i.e., with an increase of n in n-gram, the similarity scores become more effective in separating duplicates from non-duplicates. The same observations were made from using skip-gram.

5.6 Vocabulary Size and Run Time

To further evaluate the performance of different methods, we also studied vocabulary sizes (see Figure 5) and run time (see Figure 6) of different methods. We can see that, with the increase of n in n-gram (or skip-gram), both vocabulary size and run time increase exponentially. Although the performance of both n-gram and skip-gram increases with n (see Figure 4 and Table 6), both require a much larger vocabulary and therefore more RAM and longer run time. The run time of LSA-cosine is much longer than others by one to two magnitudes and therefore it is not included in Figure 6. For

Table 5: Experimental Results (ordered descendingly first by AUC and then by F1-score)

Method*	Similarity Measure	Tokeniser	Correlation	AUC	Accuracy	Precision	Recall	F1-score
OS	Overlap	Skip-gram	0.9303	**0.9952**	**0.9760**	**0.9503**	**0.9875**	**0.9686**
OG	Overlap	n-Gram	0.9312	**0.9951**	**0.9760**	**0.9503**	**0.9875**	**0.9686**
OW2	Overlap	Word-2	0.9144	**0.9949**	0.9720	0.9392	**0.9893**	0.9636
OW	Overlap	Word	0.9138	**0.9949**	0.9733	0.9469	0.9840	0.9651
TCG	TFIDF-Cosine	n-Gram	**0.9417**	0.9931	**0.9760**	**0.9566**	0.9804	**0.9684**
TCS	TFIDF-Cosine	Skip-gram	**0.9404**	0.9929	**0.9753**	**0.9549**	0.9804	**0.9675**
TCW2	TFIDF-Cosine	Word-2	0.9149	0.9910	0.9700	0.9449	0.9769	0.9606
TCW	TFIDF-Cosine	Word	0.9104	0.9899	0.9660	0.9353	0.9769	0.9556
Apollo [2]	Jaccard	5-gram	**0.9341**	0.9896	0.9686	0.9432	0.9751	0.9589
JG	Jaccard	n-Gram	**0.9330**	0.9886	0.9700	0.9374	0.9858	0.9610
JS	Jaccard	Skip-gram	0.9326	0.9885	0.9700	0.9374	0.9858	0.9610
CG	Cosine	n-Gram	0.9200	0.9885	0.9673	0.9325	0.9840	0.9576
CS	Cosine	Skip-gram	0.9192	0.9884	0.9673	0.9325	0.9840	0.9576
JW2	Jaccard	Word-2	0.9246	0.9868	0.9680	0.9401	0.9769	0.9581
JW	Jaccard	Word	0.9220	0.9864	0.9680	0.9401	0.9769	0.9581
CW2	Cosine	Word-2	0.8919	0.9861	0.9619	0.9174	**0.9875**	0.9512
CW	Cosine	Word	0.8082	0.9810	0.9499	0.8998	0.9751	0.9360
TES	TFIDF-Euclidean	Skip-gram	0.8236	0.9799	0.9579	0.9179	0.9751	0.9456
TEG	TFIDF-Euclidean	n-Gram	0.8221	0.9798	0.9579	0.9179	0.9751	0.9456
TEW2	TFIDF-Euclidean	Word-2	0.8270	0.9794	0.9593	0.9253	0.9698	0.9470
TEW	TFIDF-Euclidean	Word	0.8021	0.9765	0.9539	0.9157	0.9662	0.9403
LCW2	LSA-Cosine	Word-2	0.6981	0.9699	0.9179	0.8351	0.9733	0.8989
LCW	LSA-Cosine	Word	0.6638	0.9690	0.9126	0.8404	0.9466	0.8904
LCS	LSA-Cosine	Skip-gram	0.6301	0.9579	0.8972	0.8148	0.9395	0.8727
LCG	LSA-Cosine	n-Gram	0.6225	0.9567	0.8985	0.8233	0.9288	0.8729

* Names of the methods (except for Apollo) are derived from the acronyms of similarity measures and tokenisers (see underlines in above columns 2-3).

Table 6: Experimental Results for N-Grams and Skip-Grams (in descending order of AUC)

Similarity Measure	Tokeniser	Correlation	AUC	Accuracy	Precision	Recall	F1-score
Overlap	1-skip-4-gram	0.9402	0.9956	0.9746	0.9471	0.9875	0.9669
Overlap	1-skip-3-gram	0.9364	0.9954	0.9760	0.9503	0.9875	0.9686
Overlap	5-gram	0.9364	0.9953	0.9760	0.9503	0.9875	0.9686
Overlap	4-gram	0.9343	0.9952	0.9760	0.9488	0.9893	0.9686
Overlap	1-skip-2-gram	0.9303	0.9952	0.9760	0.9503	0.9875	0.9686
Overlap	3-gram	0.9312	0.9951	0.9760	0.9503	0.9875	0.9686
Overlap	2-gram	0.9264	0.9950	0.9746	0.9471	0.9875	0.9669
Overlap	1-gram	0.9144	0.9949	0.9720	0.9392	0.9893	0.9636
TFIDF-cosine	1-skip-4-gram	0.9478	0.9944	0.9746	0.9533	0.9804	0.9667
TFIDF-cosine	5-gram	0.9455	0.9938	0.9753	0.9549	0.9804	0.9675
TFIDF-cosine	1-skip-3-gram	0.9455	0.9937	0.9760	0.9582	0.9786	0.9683
TFIDF-cosine	4-gram	0.9441	0.9935	0.9753	0.9549	0.9804	0.9675
TFIDF-cosine	3-gram	0.9417	0.9931	0.9760	0.9566	0.9804	0.9684
TFIDF-cosine	1-skip-2-gram	0.9404	0.9929	0.9753	0.9549	0.9804	0.9675
TFIDF-cosine	2-gram	0.9367	0.9924	0.9746	0.9517	0.9822	0.9667
TFIDF-cosine	1-gram	0.9149	0.9910	0.9700	0.9449	0.9769	0.9606

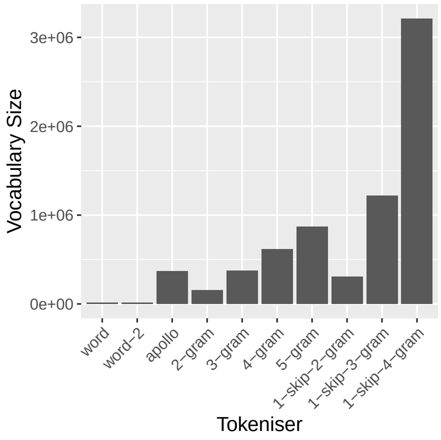

Figure 5: Vocabulary Size

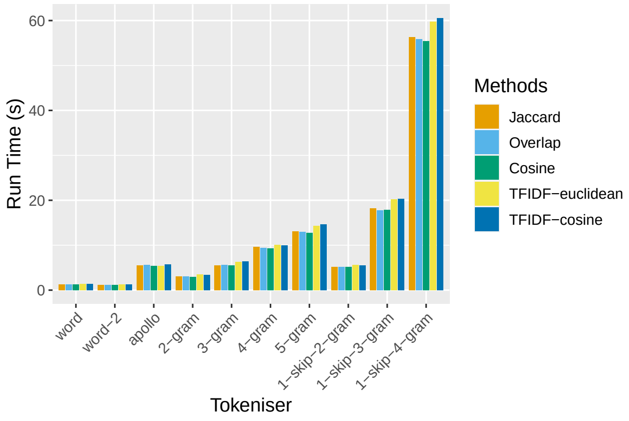

Figure 6: Run Time

example, with a 3-gram tokeniser, LSA-cosine takes 323 seconds and Overlap takes only 5 seconds.

6 DISCUSSION AND CONCLUSION

We have developed a framework for duplicate detection from online job postings, by using four tokenisers, three vectorisers and six similarity metrics and speeding it up with sliding window, blocking and re-grouping techniques. We have conducted a comparative study of all methods, together with an existing approach, using job postings from a real-world jobs board. The experiment revealed that: 1) *Overlap with skip-gram (OS)* and *Overlap with n-gram (OG)* achieved the highest AUC and F1-score, followed by *TFIDF-cosine with n-gram (TCG)* and *TFIDF-cosine with skip-gram (TCS)*, and 2) the best setting for job posting duplicat detection (in this work) is *Overlap with 1-skip-4-gram tokeniser*. The best method has been used for duplicate detection from monthly job posting data for the Skills Dashboard system at Data61, CSIRO, a national research institute in Australia.

The findings of this study are not limited to improving the usability of job boards and the quality of analytics derived from online job postings. They provide direction for other research efforts aimed at detecting duplicate content in a large body of written material, such as detecting instances of academic plagiarism and duplicate webpages.

In this work, we take the body text of a job posting as a whole piece when calculating similarity. However, different components within a job posting have different meanings and therefore can be of different importance in assessing whether two postings are similar. Further research could focus on exploring whether natural language processing techniques can be used to extract various components from a job posting and provide better input to similarity calculations. Another approach is using word embedding based methods and deep learning techniques for vectorisation. Applying the developed framework and method to duplicate detection in other domains also represents an area for future research.

ACKNOWLEDGMENTS

We'd like to thank Adzuna Australia for kindly providing data for this research and the Job Skills Dashboard Team at Data61, CSIRO for providing domain knowledge.

REFERENCES

[1] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic Clustering of the Web. In *Selected Papers from the Sixth International Conference on World Wide Web* (Santa Clara, California, USA). Elsevier, Essex, UK, 1157–1166.

[2] H. Burk, F. Javed, and J. Balaji. 2017. Apollo: Near-Duplicate Detection for Job Ads in the Online Recruitment Domain. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 177–182.

[3] Anthony P. Carnevale, Tamara Jayasundera, and Dmitri M Repnikov. 2014. *Understanding Online Job Ads Data.* Technical Report. Georgetown University. https://cew.georgetown.edu/wp-content/uploads/2014/11/OCLM.Tech_.Web_.pdf

[4] Moses S. Charikar. 2002. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing* (Montreal, Quebec, Canada) *(STOC '02)*. ACM, New York, NY, USA, 380–388.

[5] D. Deming and L. B. Kahn. 2018. Skill requirements across firms and labor markets: Evidence from job postings for professionals. *Journal of Labor Economics* 36, S1 (2018), 337–369.

[6] Susan T. Dumais. 2004. Latent semantic analysis. *Annual Review of Information Science and Technology* 38, 1 (2004), 188–230. https://doi.org/10.1002/aris.1440380105

[7] Monika Henzinger. 2006. Finding Near-duplicate Web Pages: A Large-scale Evaluation of Algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. ACM, New York, NY, USA, 284–291. https://doi.org/10.1145/1148170.1148222

[8] Brad Hershbein and Lisa B. Kahn. 2018. Do Recessions Accelerate Routine-Biased Technological Change? Evidence from Vacancy Postings. *American Economic Review* 108, 7 (July 2018), 1737–72. https://doi.org/10.1257/aer.20161570

[9] Valentin Jijkoun. 2016. Online job postings have many duplicates. But how can you detect them if they are not exact copies of each other? https://www.textkernel.com/online-job-posting-many-duplicates-can-detect-not-exact-copies/

[10] Kaggle. 2016. Avito Duplicate Ads Detection. https://www.kaggle.com/c/avito-duplicate-ads-detection

[11] Dan Lecocq. 2015. Near-Duplicate Detection. https://moz.com/devblog/near-duplicate-detection

[12] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting Near-duplicates for Web Crawling. In *Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada) *(WWW '07)*. ACM, New York, NY, USA, 141–150.

[13] Christian Thiele. 2020. *cutpointr: Determine and Evaluate Optimal Cutpoints in Binary Classification Tasks. R package version 1.0.32.* https://CRAN.R-project.org/package=cutpointr

[14] James Thurgood, Arthur Turrell, David Copple, Jyldyz Djumalieva, and Bradley Speigner. 2018. Using Online Job Vacancies to Understand the UK Labour Market from the Bottom-Up. *Bank of England Working Paper* 742 (July 2018).

[15] W. J. Youden. 1950. Index for rating diagnostic tests. *Cancer* 3, 1 (1950), 32–35.