De-duping URLs with Sequence-to-Sequence Neural Networks

Keyang Xu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA 15213
keyangx@cs.cmu.edu

Zhengzhong Liu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
liu@cs.cmu.edu

Jamie Callan
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA 15213
callan@cs.cmu.edu

ABSTRACT

Many URLs on the Internet point to identical contents, which increase the burden of web crawlers. Techniques that detect such URLs (known as URL de-duping) can greatly save resources such as bandwidth and storage for crawlers. Traditional de-duping methods are usually limited to heavily engineered rule matching strategies. In this work, we propose a novel URL de-duping framework based on sequence-to-sequence (Seq2Seq) neural networks. A single concise translation model can take the place of thousands of explicit rules. Experiments indicate that a vanilla Seq2Seq architecture yields robust and accurate results in detecting duplicate URLs. Furthermore, we demonstrate the efficiency of this framework in the real large-scale web environment.

KEYWORDS

Web Crawling; URL De-duplication; Sequence-to-Sequence Neural Network;

ACM Reference format:
Keyang Xu, Zhengzhong Liu, and Jamie Callan. 2017. De-duping URLs with Sequence-to-Sequence Neural Networks. In *Proceedings of SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080746

1 INTRODUCTION

In the World Wide Web, different URLs frequently direct users to identical or similar web pages, the differences of which only reside in the advertisements or web boilerplates displayed. This situation is known as DUST: Different URLs with Similar Text [2]. Here are some DUST example patterns from a selected website:

- Multiple URL entries: e.g., `http://test.com/question/123/how` and `http://test.com/q/123`.
- Superfluous parameters in URLs, e.g., `http://test.com/tag/mail?sort=newest` and `http://test.com/tag/mail`
- Redirection to a constant page such as Login Failure.

DUST can improve browsing experience for web users by providing multiple entries for pages. However, it poses an efficiency problem for crawlers and search engines: identical pages with different URLs are considered as distinct files, which will cause redundant efforts in crawling, indexing and searching. Two main approaches to address this problem are content-based de-duplication and URL-based de-duplication. Content-based methods [3, 9] compare actual contents between pages and use that information to determine duplications. However, these methods require pages to be downloaded first, which consumes significant bandwidth resources and does not reduce the burden of crawlers. The other line of study conducts de-duplication solely based on URLs [2, 5, 7, 8], which uses URL transformation patterns to de-dup pages. With the patterns, crawlers can judge whether two pages are duplicates by merely examining their URLs. Our work is an extension to this line.

The first URL-to-URL mapping method was studied by Bar Yossef et al. [2], which is a string substitution approach. However, it only covers limited URL transformation patterns. Further, Dasgupta et al. [5] presented a method to represent a URL as a sequence of key-value mapping and transform URLs by generating massive rewrite rules with wildcards. However, the huge number of rules makes it not applicable to a large-scale setting. Koppula et al. [7] further automated the rule generalizing process by using a decision tree model. Lei et al. [8] also adopted a pattern tree and leveraged its statistical information to generate rules. Rodrigues et al. [10] utilized multi-sequence alignment strategy to learn to rewrite rules. Despite the improvements gained, these methods are all based on explicit rewrite rules, and are limited in the following areas: (1) Rules are usually generated by string matching and regular expressions, which do not fully utilize the semantic meanings of URLs and are fragile to noise. (2) Many generated rules are of low coverage and precision; hence further heuristic filtering steps are required.

Recent success of sequence-to-sequence (Seq2Seq) neural networks [1, 11] in natural language processing bodes well for de-duping URLs. In fact, tokens in URLs are also not just simple strings, but rather contain semantic meanings. The semantic meaning of a URL can be viewed as an instruction to locate a particular web page on the website. From this point of view, two URLs that point to the same web page "have the same meaning". Transformation between such URLs can then be viewed as a translation task. Seq2Seq models enable us to build a translation model for each website quickly, without resorting to heavily engineered rules. In this work, we present a novel end-to-end URL-based de-duping framework with Seq2Seq models, where a translation model takes the place of a number of explicit rules and achieves promising results in reducing DUST. Furthermore, we report potential drawbacks of previous evaluation methods, and propose a new evaluation scheme that directly examines the impact of URL de-duping for web crawling.

2 METHODOLOGY

In summary, our goal is to train a Seq2Seq neural network that can transform an URL sequence into a canonical form. During crawling,

SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080746

we apply the model to each URL in the crawling queue to get its canonical form. We keep track of all the canonical forms to avoid re-crawling an URL with the same form.

2.1 URL Pre-processing

An URL u can be deciphered into a sequence of components, including, but not limited to, protocols, hostnames, and query arguments, tokenized by specific delimiters. For the purpose of this task, we define two kinds of delimiters, static delimiters such as "/" and dynamic delimiters which contain "?", "=" , "&", etc. Dynamic delimiters are kept, while static delimiters are discarded during URL tokenization. An example is shown as followed:

Figure 1: An example of URL tokenization, where the static delimiter "/" is removed and dynamic delimiters "?" and "=" are kept as tokens.

Function and Argument Tokens: The token vocabulary for one website can be vast in amount and is normally an open set. For example, tokens such as "123" and "mike" in the following URL:

```
http://stackexchange.com/user/123/mike        (1)
```

will be generated and added to the vocabulary for every new user. Instead of treating the token "123" as a simple string of digits, the model should better consider it as the argument after a functional token "user". Therefore, we separate tokens into argument and function tokens based on their document frequency in a large URL collection [8]. The intuition is that the document frequency of an argument token will converge to zero while the document frequency for functional tokens such as "user" and "http:" will remain stable in a large collection of URLs. For each token, we calculate the ratio of document frequency to the total number of pages and set a threshold at 0.01 for classification, as shown in Figure 2.

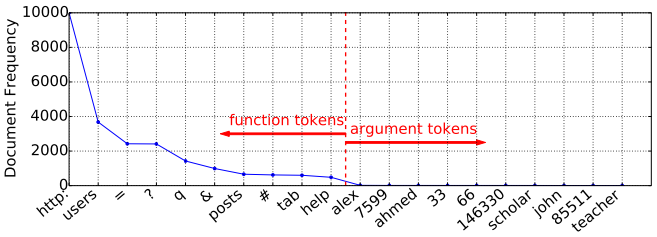

Figure 2: An example of separating argument and function tokens using document frequencies based on a corpus collected from 10,000 pages. We observe a sharp decline in the graph around the point where document frequency is 100.

Argument tokens often need to be copied exactly to the corresponding target sequence. Instead of learning to copy [6], we mimic "copying" with the following trick: each argument token is replaced with a special placeholder arg_{index}, where *index* is the n*th* argument in the URL from left. For example, URL (1) becomes

```
http://stackexchange.com/user/arg1/arg2        (2)
```

The Seq2Seq model now only learns with placeholders. Final outputs are obtained by substituting the placeholders with the original arguments. This simple replacement trick encodes positional aspects of the arguments, and is found to be effective in experiments.

2.2 Collecting Duplicating URLs

We first construct **URL duplicate clusters**, where all URLs in the same cluster refer to near-identical web pages. As in prior work, we employ a content-based clustering method proposed by Manku et al. [9]. All web pages are represented by 64-bit fingerprints. Hamming distance is utilized to evaluate similarities between two pages; the minimum threshold k for hamming distance is set as 3, following the previous setting [9]. When constructing training data, we prioritize precision over recall. To further enhance precision, we require that pages in the same cluster to share the same title.

2.3 Training Pair Construction

Given a URL duplicate cluster, the framework requires URLs in the cluster to be translated into one canonical form for matching. We can then simply train the Seq2Seq model to learn to translate all the URLs in the cluster to the canonical form. In other words, the training set should contain a list of URL pair $(\text{url}_{\text{source}}, \text{url}_{\text{target}})$, where $\text{url}_{\text{source}}$ is every URL in the cluster, and $\text{url}_{\text{target}}$ is the canonical form. We adopt a simple approach for canonical form selection: we always select the URL with the minimum number of tokens; ties are broken by selecting the URL with higher document frequency sum. Note that by doing this, we also include $(\text{url}_{\text{target}}, \text{url}_{\text{target}})$ pairs in the training instances.

To judge the quality of the generated training data, we randomly select 100 training pairs on each website and report a precision of 97.3% by human annotations. Most of errors are caused by clustering different pages that contain few main content and at the same time share the same general title.

2.4 Sequence-to-Sequence Model

We use a Seq2Seq neural network as our URL translator. An illustration of Seq2Seq model is shown in Figure 3. Both encoder and decoder use Gated Recurrent Units (GRU) [4] and share the same token embeddings. The objective is to minimize the average cross-entropy (ACE) loss, defined as follows:

$$\mathcal{L}_{ACE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} I_{ik}\log P(y_i = k|\mathbf{x}_i) + (1 - I_{ik})\log\left(1 - P(y_i = k|\mathbf{x}_i)\right) \quad (3)$$

where N is the length of decoder sequence and C is the size of vocabulary; I_{ik} denotes whether state i in decoder generates token k and I_{ik} is $\{0, 1\}$ encoded. We use Stochastic Gradient Descent(SGD) for optimization and the training batch size is 20. Our implementation is based on the open-source package Tensorflow.

We employ a Seq2Seq model that has a single layer with 512 hidden units for GRU and 512 embedding size for token vocabulary. We also explore several extensions such as bidirectional RNNs and attention mechanism, but we do not obtain significant improvements. Thus we only report results using the vanilla Seq2Seq model.

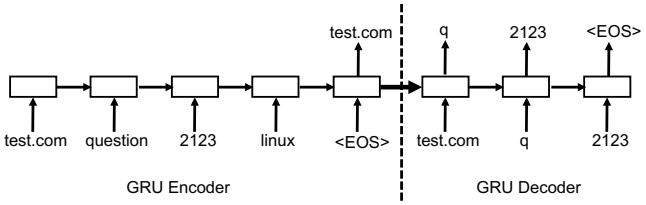

Figure 3: A Seq2Seq model [11] that reads an input URL token sequence and generates a new token sequence as the output. The model stops making predictions after outputting the token <EOS>.

After running the translator, we substitute the placeholders with the original arguments (§2.1). For example, our model will translate URL (2) as: `http://stackexchange.com/user/arg1`, Since we record `arg1` to be "123", our final translated URL can be recovered as: `http://stackexchange.com/user/123`.

3 EXPERIMENTS

3.1 Setup

Datasets: Since all URL-based methods are site-specific, in this work we construct the dataset from five websites. For each website, the first 20,000 pages with distinct URLs are crawled starting from the home page. We use the first 15,000 pages for training and the remaining 5,000 for testing, to simulate real crawling scenarios. See Table 1 for details of the five corpora.

Table 1: Statistics for five websites in our dataset. $\#C_{\text{Train}}$ and $\#C_{\text{Test}}$ are the number of clusters for 15,000 pages in training data and 5,000 pages in test data, including singleton clusters. #Train Pairs is the number of training instances, including ($\text{url}_{\text{target}}$, $\text{url}_{\text{target}}$). #Vocab is the vocabulary size of our model, after argument replacement (§2.1).

Websites	$\#C_{\text{Train}}$	#Train Pairs	$\#C_{\text{Test}}$	#Vocab
StackExchange	11125	4967	4130	335
ASP.NET	11634	3928	3467	327
Hupu	8729	12290	3696	31
Douban	10948	2574	4701	225
YouTube	14686	454	4903	88

Baselines: Two rule-based baselines are adopted, namely the Rewrite Approach (`RA`) [5] and the Decision Tree based Approach (`DTA`) [7]. Although the training data in prior works were obtained differently, we use our data to train all the methods for fair comparison. After a first round of rule generation, we filter out rules with a false-positive rate higher than 0.10 [5, 7]. We also add a naive baseline denoted as `KeepAll` for comparison, which basically crawls all URLs in test data.

3.2 Evaluations

Many of the previous work conducted evaluations using only the URL duplicate clusters. However, URLs that do not have DUST problems (which are the majority) are not included. Their evaluation metrics failed to consider such URLs. In this paper, we propose a new evaluation scheme that simulates actual crawling. For URLs gathered by the crawler, we use the de-dup model to determine whether an URL has already been crawled. Only new un-crawled URLs are crawled. We then compare the final crawled URL set with the ground truth data (content-based clusters), and compute the following metrics:

- **Precision (Prec)**: We first compute how many ground-truth content clusters are covered by our crawled URLs, denoted as $\#Covered$. A content cluster is considered as covered if any URL in the cluster is crawled. The Precision is computed as $Prec = \frac{\#Covered}{\#Crawled}$, where $\#Crawled$ is the total number of URLs crawled. A high Precision indicates better reduction ratio. To the extreme, if only one URL is crawled, the Precision is 1.0.
- **Recall (Rec)**: It measures the percentage of crawled content-based clusters: $\frac{\#Covered}{\#Clusters}$, where $\#Clusters$ is the number of content-based clusters. The recall value emphasizes amount of information captured. If all URLs are crawled, the Recall is 1.0.
- **F1 Score**: The harmonic mean of Precision and Recall.

3.3 Results and Discussions

Table 2 shows the experimental results for different methods on 5 websites, evaluated by above mentioned metrics. On average, Seq2Seq performs best on all three metrics.

The first three websites in our corpus actually contain many duplicates. On these datasets, our Seq2Seq model performs significantly better than rule-based methods on Precision. Especially for StackExchange, the rule-based method fails to generate valid rules for most of the duplicates (examples in §3.4). This demonstrates the strength of a `Seq2Seq` model: it can automatically extract transformation rules from data, which are sometimes difficult to be hand engineered. However, in the last two websites, Douban and YouTube, we do not observe a lot of duplicates from the crawled URLs. `KeepAll` actually performs pretty well on these two sets. The performance problem of `Seq2Seq` on YouTube is mainly caused by a low Recall score. During error analysis, we found that there are very few valid training pairs and the system only learns the noise from a few invalid cases.

Note that no duplication methods can have a recall higher than the `KeepAll` baseline. In practice, crawlers could select methods using a weighted F1 score, based on specific needs. For example, to emphasize higher reduction ratio, one should use a larger weight for Precision.

3.4 Case Study

Table 3 shows some examples for several types of duplicates. Here we present some correct and error cases produced by our systems.
(I) Correct Cases:

- **Superfluous parameters**: Case 1 in Table 3 shows an example, on which both rule-based and Seq2Seq methods perform well.
- **Substitutions**: Case 2 in Table 3 shows an example for which both methods work correctly. However, rule-based methods make mistakes in Case 3. Their wildcard string matching method fails to the recognize the functional token "tagged", and hence wrongly adopts a substitution pattern. However, Seq2Seq understands the semantic of "tagged" as a topic function and transforms perfectly.
- **Redirection**: See Case 4, where both methods can map all redirected URLs to a randomly selected constant page.

Table 2: De-duping URLs performance on five datasets, using two rule-based methods and the Seq2Seq model. Best results in each metric are marked bold.

Methods	ASP			Stack			Hupu			Douban			YouTube			Avg		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
KeepAll	.693	1.0	.819	.826	1.0	.905	.739	1.0	.850	.940	1.0	.969	.981	1.0	.990	.836	1.0	.907
RA	.900	1.0	.947	.827	.997	.904	.938	.995	.966	.958	.999	.978	.969	.485	.647	.918	.895	.888
DTA	.900	1.0	.947	.827	.997	.904	.996	.514	.678	.949	.999	.974	.982	.999	.990	.931	.902	.899
Seq2Seq	.949	.974	.936	.969	.905	.936	.992	.988	.990	.956	.959	.957	.991	.878	.931	**.971**	**.941**	**.955**

Table 3: Some examples of source URLs and their canonical forms. Oracle indicates the ground truth form. Due to space constraint, we replace website domains such as http://forums.asp.net by single token "domain". "√" indicates the translated form matches the oracle form. Shaded cells emphasize identical transformations.

	Source	Oracle	Rule-based	Seq2Seq
1	domain/users/150781/rich	domain/users/150781	√	√
2	domain/questions/132218/	domain/q/132218	√	√
3	domain/questions/tagged/google	domain/questions/tagged/google	domain/q/tagged	√
4	domain/private-message/set/Yovav	domain/private-message/set/anzer	√	√
5	domain/t/prev/10135	domain/t/98579.aspx	domain/t/prev/10135	domain/t/prev.aspx/10135

(II) Error Cases:

- **Database Mapping**: Case 5 shows an example of database mapping, where the previous post of "10135" is "98579" in the database. This is a very hard situation for both methods.
- **Unseen Patterns**: The best strategy for transforming unseen patterns is to entirely copy themselves. Seq2Seq performs well on most of the cases. However, if the unseen patterns contain some frequent tokens with fixed transformation, Seq2Seq tends to transform in a similar way.

3.5 Scalability

The content-based clustering for constructing training data is well designed for parallel computing and is efficient in a large scale [9]. For training the Seq2Seq model, the number of training instances is linear to the size of URL set. By separating function and argument tokens, we greatly reduce the size of vocabulary to hundreds of tokens and makes it much efficient to learn model on the site level in the real web environment.

Given a specific URL at test phase, rule-based methods need to match it against all existing rules to decide which to apply. As for Seq2Seq, the computational cost is fixed after training the model.

4 CONCLUSION AND FUTURE WORK

This paper is the first to explore a URL de-duping method using Sequence-to-Sequence neural networks. We represent a URL as a sequence of tokens and directly learn a translation model to transform it into a canonical form. Examining the uniqueness of canonical forms helps web crawlers to avoid crawling duplicates. Compared with heavily engineered rule-based methods, our Seq2Seq model is easier to build and can better understand the semantic meaning of tokens in a URL. By simulating crawling, our model achieves promising precision, recall and F1 scores in de-duping URLs. Also, our method is very efficient in a web scale because the actual functional vocabulary size is relatively small.

Currently, argument tokens are handled by a substitution trick. In the future, we plan to explore RNNs with copy mechanisms [6] to reduce the need for substitutions. Furthermore, we plan to measure the performances of our Seq2Seq-based method on a real web-scale dataset.

Acknowledgement This research was supported by National Science Foundation (NSF) grant IIS-1160862 and IIS-1302206. Any opinions, findings and conclusions expressed in this paper are the authors' and do not necessarily reflect those of the sponsors. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
[2] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the DUST: different urls with similar text. *TWEB*, 3(1):3.
[3] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 380–388, 2002.
[4] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.
[5] A. Dasgupta, R. Kumar, and A. Sasturkar. De-duping urls via rewrite rules. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 186–194, 2008.
[6] J. Gu, Z. Lu, H. Li, and V. O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.
[7] H. S. Koppula, K. P. Leela, A. Agarwal, K. P. Chitrapura, S. Garg, and A. Sasturkar. Learning URL patterns for webpage de-duplication. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 381–390, 2010.
[8] T. Lei, R. Cai, J. Yang, Y. Ke, X. Fan, and L. Zhang. A pattern tree-based approach to learning URL normalization rules. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 611–620, 2010.
[9] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 141–150, 2007.
[10] K. W. L. Rodrigues, M. Cristo, E. S. de Moura, and A. S. da Silva. Learning URL normalization rules using multiple alignment of sequences. In *String Processing and Information Retrieval - 20th International Symposium, SPIRE 2013, Jerusalem, Israel, October 7-9, 2013, Proceedings*, pages 197–205, 2013.
[11] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.