Electronic Commerce Research and Applications 40 (2020) 100961

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/elerap

Topical classification of domain names based on subword embeddings

Chong Wang[a,*], Yi Chen[b]

[a] S&P Global, 55 Water Street, New York, NY, USA
[b] New Jersey Institute of Technology, University Heights, Newark, NJ, USA

ARTICLE INFO

Keywords:
Domain names
Text classification
WWW
Internet
E-Commerce

ABSTRACT

A good domain name can help a company rapidly increase their brand awareness, attract more visitors, and therefore obtain more customers. Due to the exponential increase in the number of domain names, registrants are often frustrated because their preferred domain names are already taken. In order to enhance registrants' satisfaction and efficiency, as well as to increase the revenue of registrars (e.g. GoDaddy, Yahoo, Squarespace), it is important to suggest alternative domain names that are available. The first step is to detect registrants' needs by classifying the attempted domain name to one of the categories.

This study is the first that defines the problem of domain name classification, which classifies a registrant's preferred domain name into pre-defined categories. The paper proposes deep neural networks with subword embeddings that are built in multiple strategies. We build embeddings for character n-grams of a domain name by learning from training data, learning from external corpus, or learning from external corpus and adjusting based on training data. The experiments show that the proposed methods significantly outperform the baselines.

1. Introduction

Domain names have been at the forefront of the digital experience for decades. They provide the real estate for businesses to build their online presence and provide doorways for their customers to find them (Lodico, 2017). An eye-catching and memorable domain name adds credibility to the business and increases awareness of the brand. Thus, selecting and buying a good domain name can be the first step in attracting website visitors, generating traffic to the website, and building a reputation, which, in turn, will result in more customers and better sales.

Indeed, the domain name industry is a billion-dollar industry because of the competitiveness involved in securing the perfect domain name. Stories abound of domains that were purchased for $8 dollars 15 years ago being sold today for millions (Styler, 2015). According to a report made by VeriSign,[1] the first quarter of 2018 closed with approximately 333.8 million domain name registrations across all Top-Level Domains (TLDs), an increase of approximately 1.4 million domain name registrations, or 0.4 percent, compared to the fourth quarter of 2017. Domain name registrations have grown by approximately 3.2 million, or 1.0 percent, year over year (Verisign, 2018). In the current business ecosystem, domain name registrants submit their preferred domain names to registrars, such as GoDaddy,[2] Yahoo,[3] or Squarespace.[4] If the domain names are available, the registrants can register by paying the fee to the registrars.

With the explosive growth in the number of domain names, preferred domain names are often already registered. According to a white paper (ANI, 2019) published by Verisign, which runs the.com and.net namespaces, the third quarter of 2019 closed with 359.8 million domain name registrations across all top-level domains (TLDs), an increase of 5.1 million domain name registrations, or 1.4%, compared to the second quarter of 2019. Domain name registrations have grown by 17.4 million, or 5.1%, year over year. With the boost of the number of domain names, concerns have been raised that registering domain names gets increasingly difficult (Dunn, 2017): According to Allemann (2017), all of the good.com domains were taken and that 99% of all registrar searches today result in a "domain taken" page. Many well-known brands have encountered challenges securing their preferred domain names. For example, Google renamed their parent company "Alphabet," but it does not own alphabet.com - BMW does. In 2014,

* Corresponding author.
E-mail address: chong.wang@spglobal.com (C. Wang).

[1] VeriSign, Inc. is operating two of the Internet's thirteen root nameservers, the authoritative registry for the.com,.net, and.name generic top-level domains and the.cc and.tv country-code top-level domains, and the back-end systems for the.jobs,.gov, and.edu top-level domains.
[2] https://www.godaddy.com.
[3] https://www.yahoosmallbusiness.com/order/domains.
[4] www.squarespace.com/Domain-Names.

https://doi.org/10.1016/j.elerap.2020.100961
Received 9 March 2019; Received in revised form 4 February 2020; Accepted 11 February 2020
Available online 27 February 2020

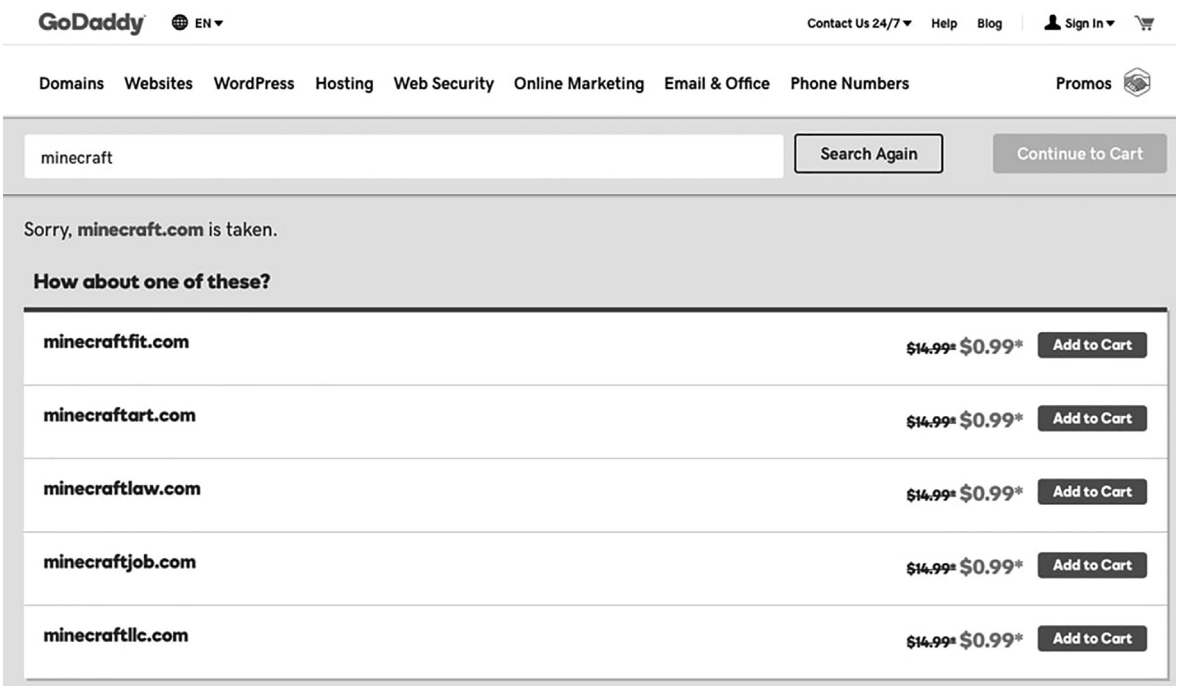

Fig. 1. GoDaddy's Domain Name Recommendation.

Microsoft spent $2.5 billion to acquire Mojang, the Swedish company behind the popular Minecraft video game, but it did not get the domain nameminecraft.com with it - that belongs to an Australia mining-engineering company (Dunn, 2017).

In the cases that the preferred domain names are not available, to assist the registrant to find an available name quickly and easily, registrars recommend alternatives that have not been taken yet, by adding a prefix and/or suffix, editing words, and/or changing the TLD of the original requested domain name. Figs. 1 and 2 are two example responses from GoDaddy and Yahoo, in whichminecraft.com is submitted as the preferred domain name. Since the domain name is not available, the two domain name recommendation systems edit the submitted domain name using general and popular words, e.g., "job", "art", "rental", "photograph", which do not seem relevant to the query. In such case, the registrant has to come up a new domain name again himself or even stop searching.

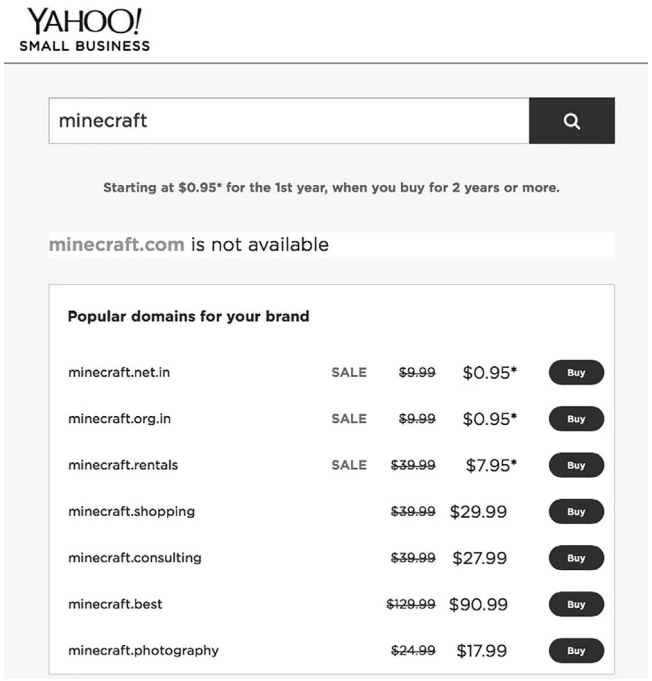

Fig. 2. Yahoo's Domain Name Recommendation.

To generate highly relevant recommendations, registrars must well understand the needs of registrants by analyzing the preferred domain names. To understand a domain name query for further transforming it, one intuitive way is to classify it into one of the pre-specified topical categories, such as business, sports or arts. The outcome can be used for downstream processors to make the preferred domain names available: For example, a set of modifiers pre-defined in the category can be used to transform the domain name. For instance, a good classifier correctly assignsminecraft.com into the "Games" category. Possible modifiers alter this unavailable domain name intominecraftgame.com,minecraftapp.com, andaddictingminecraft.com, which have not been taken. These modifiers can be created either manually or statistically based on co-occurrence or similarities (Mei et al., 2015; Levy et al., 2015; Asr et al., 2018). Therefore, as the first step, topical classification significantly impact the performance of the domain name recommendation. The risk of mis-classification can be hedged by showing new domain names from all of the top possible categories. In this case, registrants can find good and available domain names in the top part of the list. Therefore, a domain name recommendation powered by domain name classification may reduce registrants' efforts and thus boost domain name purchase.

However, accurate topical classification of domain names is highly challenging for several reasons. First of all, when a domain name is requested, the actual website is not available yet. Thus, the meta-data of the website, such as description, traffic, and registration information, are not available. The only information that can be utilized is the requested domain name itself. Second, domain names are extremely short, and thus are ambiguous in nature. Since shorter domain names are more memorable to customers, domain names are designed to be as concise as possible: most domain names have only 1–3 tokens (after word segmentation), making it challenging to identify the meaning. For example, it is hard to tell whether desktopsolutions.net is a website selling desktop furniture or providing IT services. Third, domain names usually contain noise. For example, inredpoint-design.com (a web

design and marketing website named “red point”), “red” and “point” are not related to the website topic - only “design” is. Fourth, word segmentation of domain name may introduce errors, increasing the classification difficulty. For example, the root domain part ofminecraft.com may be segmented to “mine” and “craft”. Finally, some domain names are almost impossible to understand, e.g.,abcdefg.com, 843625.org, or carlwang.net. These domain names do not contain valid or meaningful English words. To the best of the knowledge, there is no existing study that attempts to understand the topics of domain names. The industry is still looking for good solutions.

To accurately classify domain names, we propose utilizing the subword (i.e., character n-grams) information of the domain names. With well-represented subword semantics, we propose deep neural network-based models. In particular, the input of the models is a domain name which is segmented into tokens. The subword information of each token is modeled by latent vectors (i.e., embeddings). The subword embeddings are aggregated to represent the semantic of each token. Fed with token semantic representations, domain name semantic representation, and the TLD, a Recurrent Neural Network (RNN) (Tang et al., 2015) and a Convolutional Neural Network (CNN) (Zhou et al., 2018) are adopted to predict the category label, respectively. The output is the predicted category of the input domain name. In the experiments, we evaluate the proposed methods and compare three different methods of building subword embeddings. The experiment results show that the CNN, with frozen embeddings learned from Wikipedia, obtains the best performance in both classification and ranking tasks.

With our models, to make recommendation based on these domain names, in practice, a recommender system can rank the possible categories of a user-submitted domain name by the predicted probabilities. The recommendation list can include recommended domain names from all of the first several highest categories. For example, the predicted probability distribution of “desktopsolutions.com” is 60% for Computer and 40% for Home. Modifiers in both categories alter this unavailable domain name into a list of available domain names. These new domain names can be ranked by the product of the confidences of each modifier and the predicted classification probabilities. Hence, the recommendation result can consist of new domain names from both categories. Those from the Computer category might be ranked higher than those from the Home category.

This work will benefit both users and firms on the sell side of the domain name registration. Users can receive more relevant alternative domain names. The firms can obtain more revenue by selling more domain names. In addition, the proposed method can also be applied in phishing website detection (given the assumption that whether a website can be malicious can be partially told by its domain name). Being fed by a large scale of labeled data on phishing detection, the proposed model can classify domains into one of the two categories: Malicious or Benign.

The contributions of our paper are as follows: 1) We define a new research problem, topical classification of domain names. It can significantly benefit domain name recommendation, currently a billion-dollar industry: A user-submitted domain name is classified by its topic. A set of domain modifiers specific to each category can be then triggered to alter the original domain name. As a result, a topical classifier of domain names can suggest available domains that users may would like to purchase. A good solution may reduce users’ effort to find available domains and boost domain name registrars’ revenue. 2) Due to the polymorphism and noisiness in domain names, we propose to use character-level RNN-based and CNN-based models, which have been used in text classification tasks. To better capture the semantics of domain names, we propose to compute abstract representations for both the domain names tokens and the entire domains. 3) We evaluate our models on a publicly available dataset. The results show that our models significantly outperform the baselines. 4) Besides domain name recommendation, the proposed method can also benefit other fields, such as Internet security, in which malicious websites can be detected from the links and domains.

The rest of the paper is organized as follows: Section 2 compares against related work, Section 3 introduces the problem definition, Section 4 presents the proposed methods, Section 5 shows extensive experimental results, and Section 6 concludes the paper.

2. Related work

2.1. Topical classification of URLs

To the best of our knowledge, none of the existing work focuses on topical classification of domain names. Several existing studies attempted to classify Uniform Resource Locators (URLs) into categories in two main applications.

One of the main applications of URL classification is to detect malicious or phishing websites based on their URLs without accessing the content of Web sites. Thus, it eliminates the run-time latency and the possibility of exposing users to the browser-based vulnerabilities. The work (Ma et al., 2009; Liu et al., 2016) describes approaches to this problem based on automated URL classification, using statistical methods to discover the lexical, site popularity features, and host-based properties of malicious Web site URLs. Le et al. (2011) try to discover phishing websites that steal personal user information and pretend to be legitimate. The authors automatically select lexical features and hand-select obfuscation-resistant features. They also utilize external feature collections, e.g., WHOIS and Team Cymru, in order to get registration information, the network information, and the geo-location of each URL. Huang et al. (2014) conduct a bottom-up search method and a greedy selection algorithm that iteratively discover patterns in the malicious URLs. Saxe and Berlin (2017) propose a character-level Convolutional Neural Network with embeddings to detect malicious URLs, file paths, and registry keys. They build one embedding for each character and then stack CNN layers on top of the embedding layer. The output is whether a URL is benign or malicious.

The other application is identifying the topic of a webpage merely based on its URL. Baykan et al. (2011), Abdallah and de La Iglesia (2014), Rajalakshmi and Aravindan (2018) extract textual features from a given URL: tokens, n-grams from tokens, n-grams from URL, and/or n-grams with positions. A classifier, e.g., Support Vector Machines or Language Model, is built by feeding it the features of training URLs. Hernández et al. (2014) classify web pages based on URLs in an unsupervised way. The proposed algorithm takes the URL of a web page with a keyword-based search form as input, and outputs a set of patterns that represent the URLs of pages that belong to the same class.

In contrast to the existing work of URL classification, domain name classification has several characteristics as following:

- Compared to a domain name, which is a just part of a URL, a URL contains more information about a web page: A URL has more text in the sub-directory part (refer to Fig. 3). The sub-directory part of a web page usually contains the title and/or topical channel of the

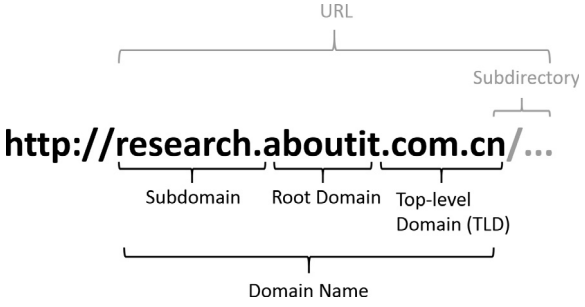

Fig. 3. Terminology.

web page. Conversely, domain names are much shorter than URLs. According to the statistics from our dataset, most domains contain 2 to 3 word tokens. It is highly challenging to identify the web page topic. Traditional lexical features (Baykan et al., 2011; Abdallah and de La Iglesia, 2014; Rajalakshmi and Aravindan, 2018; Ma et al., 2009; Liu et al., 2016), including n-grams and tokens, may not work well due to much higher sparsity.

- URLs have significant patterns, especially URLs under the same web site. For example, a web page about news may follow patterns, likewww.forbes.com/sites/<authorname>/<date>/ (Huang et al., 2014; Hernández et al., 2014). These patterns can be revealed and utilized for classification. However, it is impossible to discover patterns from domain names. Since domain names cannot be duplicated, a desired but unavailable domain name has to be transformed (includes swapping, replacement, interpolation, and deletion) arbitrarily in order to be registered. Hence, common patterns may not exist within domain names in the same category.
- In the application of domain name recommendation, users submit the domain names that they want to register. Since the web sites with the domain names do not exist yet, external domain name features utilized in Le et al. (2011) (e.g., popularity, registration, geo-location, and network information) are not available. Thus, the domain name itself is the only input that can be used for classification.

2.2. Deep neural networks for text classification

Extensive studies have proposed to applied deep neural networks to solve natural language processing problem. Based on how the input words are processed, existing studies can be categorized in three groups: word-level, character-level, and hybrid.

Kim (2014) proposed a simple CNN with little hyper-parameter tuning and static word embeddings achieves excellent results on multiple benchmarks. Instead of using a word-level CNN, Dos Santos and Gatti (2014) proposed a character-level CNN that learns character-level embeddings which were used to compute word-level and sentence-level representation. They stated that using character-level embedding can save more memory and handle out-of-vocabulary words. To further take word order into account, Johnson and Zhang (2014) studied CNN on text categorization to exploit the word order of text data for accurate prediction. There are also some studies attempted to use both word-level and character-level representations. For example, Santos and Zadrozny (2014) proposed a deep neural network that can learn character-level representation of words and associate them with usual word representations to perform POS tagging. To enable transfer learning, Severyn and Moschitti (2015) used an unsupervised neural language model to train initial word embeddings that were further tuned by the proposed CNN model. Tang et al. (2015) and Tang et al. (2015) introduced a RNN based model to learn vector-based document representation in a unified, bottom-up fashion. They learned word embeddings directly from the training data for sentiment classification. On the other hand, for character-level embeddings, Zhang et al. (2015) and Zhang and LeCun (2015) offered an empirical exploration on the use of character-level convolutional networks (ConvNets) for text classification. They showed that character-level CNN could achieve state-of-the-art or competitive results. Ma et al. (2015) combined deep learning techniques and traditional NLP (i.e., dependency tree). They proposed a very simple dependency-based CNN. They considered a word and its parent, grand-parent, great-grand-parent, and siblings on the dependency tree. Wang et al. (2016) described a jointed CNN and RNN architecture, taking advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN for sentiment analysis of short texts. Kim et al. (2016) employed a CNN and a highway network over characters, whose output was given along short-term memory (LSTM) recurrent neural network language model (RNN-LM). The authors claimed the model outperformed word-level/morpheme-level LSTM baselines. Liang et al. (2017) proposed a hybrid model, combining the merits of word-level and character-level representations to learn better representations on informal text.

Generally, word-level representations directly capture word semantics from training corpus. However, it requires a large vocabulary. It is not able to handle out-of-vocabulary words. On the other hand, character-level representation can handle unseen words. It is also insensitive to word morphology. However, it may have large computational cost, as character-level representations have to be aggregated to the word-level.

Deep learning has been used in many NLP applications, such as news classification, sentiment analysis, and machine translation. However, as far as we know, there is no existing studies on topical classification for domain names. In our study, as a domain name is no longer available for registration once it is ready taken, many domain names contain variations, abbreviations (e.g., “med” for “medicine”), fillers (e.g., “abc”, “123”), and/or word stems (e.g., affixes). Therefore, relying on word-level representation leads to low classification performance due to out-of-vocabulary words and noise. In addition, domain names are extremely short. Most domain names have only 1–3 tokens. Thus, it is not meaningful to use more advanced models, such as Long Short-Term Memory (LSTM), attention (Vaswani et al., 2017), ELMo (Peters et al., 2018), designed for long input sequences. Due to the polymorphism and noisiness of domain names, we propose to use character-level representations to capture the semantics of domain names. Character-level representations that consider subwords can alleviate the impact of word variations, abbreviations, and word stems.

3. Problem definition

3.1. Terminology and problem definition

There are multiple systems of terminology for URLs. To make it clear, below is the one used in this paper. A typical URL contains a **subdomain**, a **root domain**, a **top-level domains (TLD)**, and subdirectories. A TLD may consist of multiple suffixes, such as “co.uk” or “info.us”. The combination of the first three parts is called a **domain name**, or a **domain**. In this work, we focus on topical classification of domain names.

Problem Definition. Given a domain name, e.g., www.foxnews.com, our goal is to assign it into one of the topical categories, e.g., sports, reference, news. In this project, these categories are pre-defined in the dataset we use.

Domain names are first segmented by dots among subdomains, root names, and top-level domains (TLD).

Subdomains and root domains may consist of multiple words which are sometimes concatenated without any delimiters, e.g., “aboutit”. Thus, a non-trivial method should be applied for word segmentation. The segmented words are referred to **tokens** in this paper. For instance, the subdomain and root domain in Fig. 3 are segmented into “research”, “about”, and “it”.

Besides tokens, our proposed model also takes **subwords** into account. Subwords are character n-grams (**char n-gram**) of a word. For example, the 5-grams of “research” are “resea”, “esear”, “searc”, and “earch”.

3.2. Dataset

In this project, we use a publicly available dataset, i.e., the Open Directory Project (ODP) DMOZ dataset,[5] which has been used in many existing studies (Baykan et al., 2011). The dataset organizes web page URLs into 15 first-level categories. Within each first-level category, webpages are organized based on a hierarchical structure. In this

[5] http://dmoztools.net.

project, we only focus on the first-level categories. It is observed that the web pages in the regional category are also in other categories and the pages in the world category are not in English. Following the practice of Dai et al. (2006), we remove the regional and world categories (13 categories are left). Note that DMOZ data is on the semantics of URLs, not domains. A domain could have URLs that belong to different categories. To obtain a reliable gold standard, we remove domains whose URLs are in more than one categories. After de-duplication, we obtain about 200 K domains in total.

4. Proposed methods for topical classification of domain names

4.1. Empirical observations

As discussed earlier, topical classification of domain names is highly challenging. To address it, we have several observations.

First, domains are extremely *short*, with only 1–3 tokens. Compared with traditional short text classification, it is non-trivial to model topic (e.g., using Latent Dirichlet Allocation (LDA) (Wang et al., 2018) and/or utilize the relationship between words (Watrous-deVersterre et al., 2012). Thus, it is important to utilize all information provided in the domain names.

Second, domain names are *polymorphic*. To avoid duplicates and/or keep it short, people may come up with the domains by using abbreviations and/or different forms of the words, e.g., “medicine” to “med”. The resulted tokens may not be valid English words, but they could be relevant to its topic. Therefore, the proposed solution should take morphological information into account.

Third, top-level domains (TLD), e.g., com, edu, and org, sometimes indicate the topics. For instance, “columbia.edu” is the official website of Columbia University, while “columbia.com” is linked to a sportswear company. The TLDs can help distinguish them.

4.2. Subword embedding learning

Due to large variability in domain names, we propose to capture subword semantics, instead of directly capturing the meaning of a token as a whole. The main reasons are 1) Domain names usually contain abbreviations. A number of abbreviations are subwords of the original words, e.g., “med” is short for “medicine”. Directly building embeddings for entire words may not be able to handle unseen abbreviations in test data. 2) Registrants often attach affixes to a word stem to form an available domain name. Using subword embedding can reduce the impact of these affixes. The meanings of subwords are filtered and aggregated for the entire token.

However, it is not evident how subword embeddings should be combined to a token. Surprisingly, simply averaging word embeddings of all words in a text has proven to be a strong feature across a multitude of tasks (Kenter et al., 2016; Yu et al., 2014; Gershman and Tenenbaum, 2015). The method of averaging subword embeddings is also adopted in *fastText* (Joulin et al., 2016). Thus, we use averaging to aggregate subword embeddings to token embeddings.

Subword embeddings have been proven to be effective in fastText. FastText represents the meaning of words by unsupervisedly learning low-rank vectors from a large textual collection. The vector of a word is obtained by summing vectors of the char n-grams appearing in the word. In this case, unlike Word2Vec (Mikolov et al., 2013), as long as their char n-grams occur the textual collection, fastText can “predict” the semantic vector of unseen words, i.e., embeddings, and the character sequences that are not valid English words. Thus, fastText can alleviate the issue of polymorphy.

In this project, we investigate several ways to learn subword embeddings and evaluate them empirically.

The first method is to initialize the subword embeddings as what we do for other parameters. Random initialization (Sutskever et al., 2013) is a commonly-used method: The weights are initialized very close to zero, but randomly. This helps in breaking symmetry and every neuron is not performing the same computation. The embeddings can be learned during the classification model training. The advantage is that the embeddings should be more aligned with the prediction target. On the other hand, embeddings may be overfit if training data is not sufficient. In our empirical evaluation, an RNN model and a CNN model with such randomly-initialized subword embeddings are denoted as *RNN-random* and *CNN-random*, respectively.

The second approach to learn subword embeddings is to train the model on an external corpus. A model is trained on a Wikipedia dump (Foundation, 2003) with a default model as proposed in Joulin et al. (2016). Given a token, the model generates a 300-dimensional embedding vector to represent its meaning in real-time. We used skip-gram to get pre-trained subword embedding. The values in the vector are not changed during training once being learned from the external corpus. The advantage of this method is that external corpus can help enhance the training data of the original task, especially when the task training data is insufficient. External corpus may contain much more unique words that are unseen in the training data. However, this method assumes that the semantics of a word learned from the external corpus still holds in our domain name dataset. Otherwise, the performance will be discounted. In the experiments, an RNN model and a CNN model with such frozen subword embeddings are denoted as *RNN-frozen* and *CNN-frozen*, respectively.

The third method is to use the embeddings that are learned from the external corpus in the previous method to initialize the subword embeddings, and then further update the embeddings during model training. In the experiments, an RNN model and a CNN model with such initialized subword embeddings are denoted as *RNN-initial* and *CNN-initial*, respectively.

Since each method has pros and cons, it is not clear which one best fit our application. Thus, in the experiment, we evaluate all three strategies by comparing the end performance of classification built upon them.

4.3. Domain name segmentation

To extract the subwords, domain names need to be segmented. Domain names are first segmented into a list of tokens. For each token, we extract the subwords.

A domain name may contain a combination of numbers, letters, and hyphens. It is also allowed to use multiple instances of hyphens, but not a double hyphen. Other forms of punctuation, symbols or accent characters cannot be used. Therefore, a domain can be first segmented by periods and hyphens. For instance, the root domain part ofhttp://www.foxnews-global.com is first segmented into “foxnews” and “global”. Further segmentation is then applied to each of the initial tokens. For instance, “foxnews” is further segmented into “fox” and “news”.

Word segmentation is an essential natural language processing step in the languages (e.g., Chinese and Japanese) wherein words are not delimited by spaces. A common practice is training a statistical model to decide where the word boundaries are. In this project, we use a publicly available API, wordsegment (Jenks, 2018), for English word segmentation. The API trains a language model (Norvig, 2009) (i.e., a probability distribution over all the n-grams in English) and learn the parameters of the model from the Google Web Trillion Word Corpus (Brants and Franz, 2006), then the model to define the probability of each candidate segmentations. The candidate segmentation with the highest probability is the final segmentation. For example, for a root domain “aboutit”, “about it” is a more probable one than “a bout it”. Thus, “about it” is the final segmentation. “about” and “it” are the tokens in the root domain.

Each token of a domain name is then split into subwords (i.e., char n-grams). The semantics of the subwords are represented by subword embeddings discussed in Section 4.2. Taking Fig. 3 as one example, the

TLD “com.cn” is stripped and forms a separate feature. “research.aboutit” is then segmented into tokens: “research”, “about”, and “it”. Subwords are then extracted from each token. Assuming $n = 3$, the subwords of “research” are “res”, “ese”, “sea”, “ear”, “arc”, and “rch”. The subwords of “about” are “abo”, “bou”, and “out”. The subwords of “it” is “it”, since its length is less than three. The embeddings of the six subwords in “research” are used to model the semantic of “research”. Similarly, the embeddings of the three subwords in “about” are used to model the semantic of “about”. Finally, the embedding of “it” is used to model “it”.

All subword embeddings of a domain name is fed into a neural network based model. The model classifies the domain name into one of the topical categories. This paper propose two types of models: RNN-based (Section 4.4) and CNN-based (Section 4.5).

4.4. An RNN-based method

An RNN is a class of neural networks that can handle sequences with variable lengths. Mimicking human reading behavior, RNN reads through the input text from left to right and finally generates a low-rank vector to represent the semantics of the entire input text.

Fig. 4 shows the structure of the RNN-based method. Assuming the given domain has N tokens, each token can be divided into char n-grams. For each token, we first look up the embedding of its char n-grams and then averages them together as the embedding of the entire token. An RNN reads the token embeddings from left to right and summarizes the embedding by the hidden layer of the last time step. Meanwhile, the top-level domains (e.g., “.edu”, “.org”, “.uk”) are represented by one-hot encoding: The vector is length of the number of unique top-level domains. All values are 0 except the one represents the corresponding top-level domain is 1.

In addition, it is observed that the segmentation module may mistakenly segment domains because domains may contain non-regular English words and/or unseen words. For instance, in the example of Fig. 2, “minecraft” may be unexpectedly segmented into “mine” and “craft”, which deviates its original meaning. Also, “golang”, which can be the name of a programming language, is split into “go” and “lang”. This causes a domain such as “golang.org” to be incorrectly assigned to a category other than “Computers”. To overcome this problem, besides tokens, we also add the subword embeddings of the entire domain into the model. It is observed that the subword embeddings model can capture the meaning of “golang” well and classify “golang.org” correctly.

Due to the sparse semantic carried by domain tokens, multiple fully connected layers are stacked to better abstractize and enrich the features. To alleviate overfitting, we add a dropout layer after each fully connected layer. Dropout (Srivastava et al., 2014) is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. it ignores units (i.e. neurons) during the training phase of a certain set of neurons which is chosen at random. Experimentally, we use three 300-neuron layers with 50% dropout rate (i.e., 50% of randomly-picked units are set to be 0 during the training phase). At last, a softmax function is used to map the logits into a probabilistic distribution. Each number represents the probability the domain belongs to the corresponding category.

4.5. A CNN-based method

It is also observed that domain names are usually *noisy*. Organizations tend to add their business name to their domains. For instance, “healthpointplus.com” (HealthPoint Plus) can be segmented to “health”, “point”, and “plus”. However, only the first token is relevant to its topic. The other two can be regarded as noise. In other words, many tokens in this domain are irrelevant. In addition, domain names cannot be duplicated. Therefore, to avoid duplicates, some may add irrelevant or generic but easy-to-remember character sequences in their domains, e.g., “abc”, “world”, and numbers. These filler tokens can appear as either prefixes or suffixes. Hence, models, like RNN, which considers every piece of information and considers token order, may unnecessarily learn too much noise, which hurts end performance.

Therefore, a CNN-based method is proposed to overcome this issue. A Convolutional Neural Network (CNN) (Zhang et al., 2015) consists of convolutional layers and pooling layers. A convolutional layer defines a set of learnable filters, each of which computes the output of neurons that are connected to local regions, e.g., word n-grams, in the input. The pooling layer will perform a downsampling operation by taking the max or average among the output neurons. Thus, instead of capturing the meaning of the entire domain, CNN concentrates on the local information and discards the local regions which are not sufficiently significant. For example, a root domain “UniHealthFoundation” in the health category has tokens “uni”, “health”, and “foundation”. CNN may give the largest weight to “health” than “uni” and “foundation”. CNN also does not consider the entire order of the tokens in a domain. Instead, it considers the order of tokens in local regions. In our application, CNN would be able to focus more on significant tokens.

Fig. 5 shows the structure of the CNN-based method, assuming the input domain has N tokens. Since domains are short, the filter sizes we adopt are one and two. Through experiments, we adopt 512 filters for each filter size.

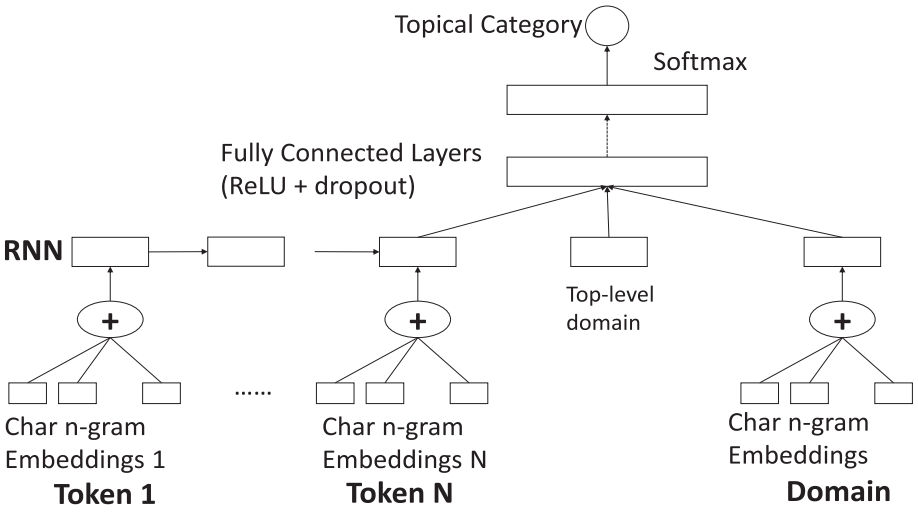

Fig. 4. The structure of RNN-based method.

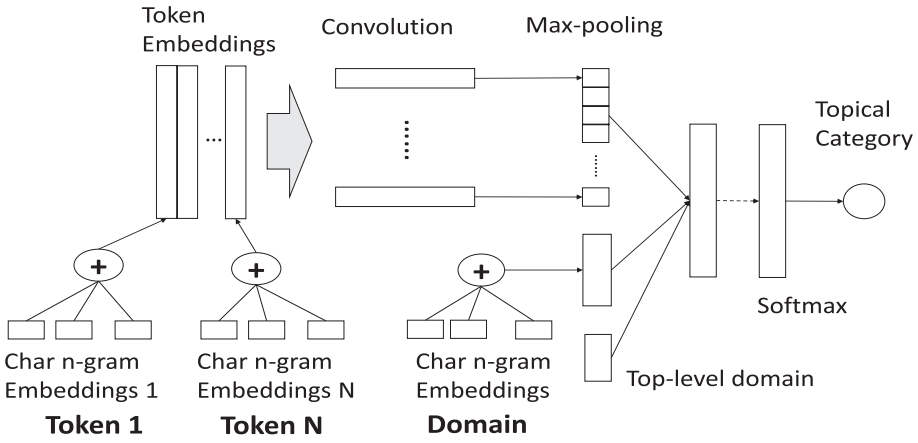

Fig. 5. The structure of CNN-based method.

5. Experiments

5.1. Experimental dataset

As introduced in Section 3.2, we use a publicly available dataset ODP. After data pre-processing, about 200 K unique domains are obtained. The largest three categories are 1) Business: 27 K, 2) Society 15 K, and 3) Arts: 12 K. The smallest three categories are 1) News: 0.72 K, 2) Home: 1.3 K, and 3) Games: 1.8 K.

90% domain names are assigned to the training set, while the rest of the domain names are randomly and equally assigned to the validation and test set. In other words, the ratio of training, validation and test sets are 18:1:1.

The validation set is used in the early stopping strategy in order to avoid overfitting: At the end of each epoch, the model obtained in the epoch is evaluated on the validation set. The model with the highest F-score is stored on disk. The total number of epoch is set to be 60. After 60 epochs, the model on the disk got the best performance. It is finally used to predict the test dataset. The experimental results are reported by threefold cross-validation.

5.2. Implementation details

Classification models are implemented in Python. All neural networks are implemented in TensorFlow. The training process is accelerated by NVIDIA GeForce GTX 1060. The CPU processor is Intel Core i7. The RAM is 32 GB.

We experimented with different parameters for the deep neural networks. Below are the parameter combinations that lead to the best classification performance. The RNN model outputs 300 hidden neurons. The CNN model has one convolutional layer with 512 filters and one max-pooling layer. The dropout rate and the learning rate of both kinds of models are 0.5 and 0.001, respectively. We use three fully connected layers, each of which has 300 hidden neurons.

The models are trained through 40 epochs with batch sizes of 512. Early stopping is adopted to avoid overfitting. The model snapshot with the lowest validation log-loss is stored on the disk. The test dataset is only predicted using the best model found on the validation data.

5.3. Comparison systems

The variants of the neural network-based methods are compared with several baselines below:

- *Token*-based method: A given domain name is segmented into tokens. All unique tokens are used to represent the domain name. It is similar to the bag-of-words model. A Support Vector Machines (SVM) model is used for classification.
- *Char n-gram*-based method: Adopts the existing method, proposed in Baykan et al. (2011) for URL classification, to classify domain names. This method can represent most existing textual approaches. In particular, below features are extracted from a domain name: 1) *Tokens*: Each domain name is lowercased and split into a sequence of strings of letters at any punctuation marks, numbers, or other non-letter characters. 2) *N-grams from Tokens*: A domain name is first split into tokens. Then character n-grams are extracted from each token. 3) *N-grams from Domain Name*: The given domain name is not split into tokens. N-grams are extracted from the entire domain name. 4) *Encoding positional information*: We duplicated each n-gram (or token) and appended its position in the domain name to it.

Through experiments, we consider 4–5-6–7-8-grams, which lead to the best performance.

5.4. Classification performance

We first conduct experiments to evaluate the performance of the proposed classifiers. Since the size of each topical category significantly varies, we use precision, recall, and F-score as the evaluation metrics. In addition, we also use accuracy (with 0.5 as the decision threshold) to give us a more intuitive indicator of the classification performance. The precision is the fraction of the test domains labeled that are correctly labeled. The recall is the fraction of the domain names in a category are that correctly labeled. F-score is the harmonic mean of precision and recall.

Table 1 is the comparison of the classification performance. We first calculate the precision, recall, and F-score of each category and find their unweighted mean. The accuracy is directly computed across categories. The results are sorted by F-score in ascending order.

According to the result, the token-based method performs the worst because it only considers tokens extracted from tokens. As a result, its feature vectors are highly sparse. Domains have little chance to share the same tokens. The char n-gram based method performs much better; it uses one-hot encoding to model subword information. In contrast to

Table 1
Classification Performance.

Approaches	Precision	Recall	F-score	Accuracy
Token	0.5390	0.3934	0.4379	0.4997
CNN-random	0.5062	0.4302	0.4532	0.5207
Char-ngram	0.5442	0.4235	0.4616	0.5230
RNN-frozen	0.5297	0.4422	0.4709	0.5322
CNN-initial	0.5280	0.4477	0.4745	0.5246
CNN-frozen	0.5600	0.4696	0.5002	0.5515

the token-based method, it can better identify domain names which share the same words with variants. As mentioned previously, word variants are common in domain names because two domain names cannot be the same. Identifying domain names with the same word stem is highly important.

We initialize the subword embeddings in different ways: 1) "random": The embeddings are randomly initialized; 2) "initial": The embeddings are initialized by the char-gram embeddings learned from the Wikipedia dump. The embeddings are adjusted during model training; 3) "frozen": The embeddings are initialized by the char-gram embeddings learned from the Wikipedia dump. They are frozen during model training.

The result shows that, with frozen embeddings, RNN-frozen is not as effective as CNN-frozen. The reason is that the output of RNN takes all char n-grams into account. However, the signal-to-noise ratio of domain names is often low. For instance, chances are one out of two that a word in a domain name is irrelevant to the topic of the website. Thus, RNN may mistakenly include noise. In contrast, the pooling layer in a CNN is able to ignore the least significant signal.

The CNN with frozen subword embeddings performs the best. Randomly initializing the subword embeddings does not lead to good performance. The reason is that it is highly difficult to learn char n-grams semantic representation completely from scratch on the domain name dataset because there are tons of unique subwords. The ODP dataset is not large enough for neural networks to fully capture subword semantics. On the other hand, Wikipedia contains subwords with sufficient occurrence. Thus, embeddings built upon Wikipedia can capture meanings more accurately. Also, Wikipedia covers more unique subwords so that "CNN-initial" and "CNN-frozen" have less unseen subwords in the test data. In addition, it is interesting to see that freezing the subword embeddings learned from Wikipedia outperforms making the embeddings trainable, i.e., "CNN-initial". To minimize the log-loss on the training data, the deep neural networks may overfit and thus deviate the subword embeddings that are learned from Wikipedia.

We also present the impact of different filter sizes, the numbers of hidden neurons in the fully connected layers, and the dropout rates.

We first vary the filter sizes of the proposed CNN models from 128 to 1024. Similar to other parameters in deep neural networks, the filter size is trade-off between under-fitting and over-fitting: The larger the filter size is, the higher complexity of the model has and the more likely it is that the model overfits the training data. The results in Fig. 6 reflect that the filter size impacts the F-score slightly. All models obtain the best performance when the filter size is 512. A too large or too small filter size may hurt the performance due to under-fitting or over-fitting. We also find that it takes 10% longer time to train a model with a filter size of 1024 than a model with a filter size of 128.

We then vary the number of hidden neurons in the fully connected layers from 100 to 500. The results in Fig. 7 show that the number of hidden neurons has slight impact on the F-score. Although wider/narrow fully connected layers may cause over-/under-fitting, it is

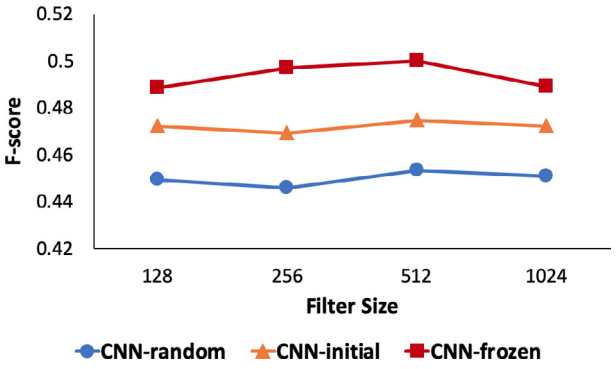

Fig. 6. Varying The Filter Size of CNN.

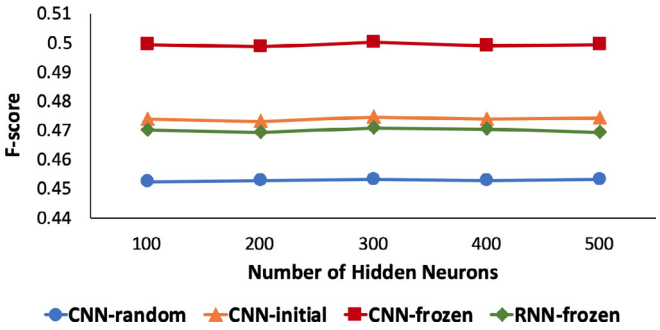

Fig. 7. Varying The Number of Hidden Neurons.

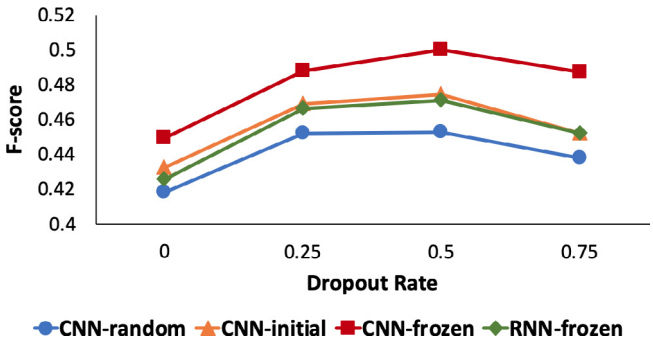

Fig. 8. Varying The Dropout Rate.

interesting to see that the F-scores at 100 and 500 are even slightly higher than those at 200 and 400. The performances of different numbers of hidden neurons are not significantly different.

We finally try different dropout rates from 0% (i.e., no dropout) to 75%. The higher the dropout rate is, the stronger regularization the model has. The results in Fig. 8 present that adding dropout layers can largely improve F-scores. Compared with the filter size and the number of hidden neurons, the dropout rate has much larger impact on the performance. A too high dropout rate (i.e., 75% in our case) tends to make the model underfit the training data: the abstract features learned in the hidden layers are randomly erased to zeros.

5.5. Ranking performance

In some business scenarios, accurately classifying a domain name into one single category may not be necessary. For instance, given a user input domain name, a domain name recommendation system may not return available domain candidates from only one category, but candidates from multiple categories. As accurately classifying a domain name is highly challenging, enhancing recommendation diversity can minimize the risk. For example, a user submitsdesktopsolutions.net. It is difficult to determine whether it should belong to "Home" or "Computers" category. Thus, we can generate top candidate domains names (that are available to register) from all possible categories and rank these domains by the probabilities that the domain belongs to the category. For example, in the return list (similar to the interfaces shown in Fig. 1 and 2), the first three candidate domains are from the "Home" category and the next four are from the "Computers" category. In this case, even though the top predicted category is incorrect, the user still can find relevant recommendations in the rest part of the ranking results.

Therefore, besides classification metrics, we also evaluate our models using ranking metrics: Mean Reciprocal Rank (MRR) and Average Rank. The mean reciprocal rank is a statistical measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the

Table 2
Ranking Performance.

Approaches	MRR	Average Rank
Token	0.6499	2.8492
Char-ngram	0.6683	2.7265
CNN-random	0.6721	2.5729
CNN-initial	0.6756	2.5369
RNN-frozen	0.6869	2.4068
CNN-frozen	0.6983	2.3786

first correct answer. Eq. (1) is how we calculate MMR: D is the number of test domain names, while $rank_i$ is the rank of the true category in the prediction of the i test domain name.

$$MRR = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{rank_i} \tag{1}$$

Since MRR only cares about the single highest-ranked relevant item, it is suitable for our problem where only one out of 13 categories is the true label of a given domain name. The higher the MMR, the better the model. In addition, as a more intuitive metric, average rank is the mean of the true category in the prediction of all test domain names. Therefore, the lower the average rank, the better the model. A classifier with an accuracy of 1 has an average rank of 1. The worst classifier has an average rank of 13 (13 categories in total).

Table 2 show the comparison result by MRR and average rank. Likewise, the CNN-frozen has the best performance in terms of both metrics. It is also interesting to see that, although the F-score of RNN-frozen is slightly lower than that of CNN-initial, the ranking performance of RNN-frozen is significantly better. This is because, even if a true category is not ranked first in the prediction of RNN-frozen, it is usually ranked in the top of the list. In contrast, CNN-initial may assign more correct labels to the test domain names. However, once it makes a mistake, the true category tends to be ranked at a lower place. This is not preferable in a real-world business scenario. For example, for domain name recommendation, it is expected that even if the correct recommendations are not in the top positions, they should be in the first page in order to have good visibility.

5.6. Detailed performance of the best classifier

Table 3 is the classification performance of CNN-frozen in individual classes. The last column is the number of test domain names in each category. Since the training and test data are split proportionally, a category with large test set also has a large training set.

It is interesting to see that the performance in each category is correlated with the size of the category. The Pearson's R between the count and the recall is 0.755. This is a strong positive correlation, which means that large category size goes with high recall scores (and vice versa). In addition, there is a weak negative correlation between category size and precision scores (Pearson's R = −0.0957). This is reasonable because a classifier may learn larger categories better and tend to assign more large category labels. Thus, the recalls of large categories are high. In this case, their precision may be inevitably hurt. As a result, the Pearson's R between F-scores and the category sizes is 0.6094. This is a moderate positive correlation. Hence, generally speaking, the classifier performs well in large categories.

Table 3
Classification Performance of *CNN-frozen* in Individual Classes.

Category	Precision	Recall	F-score	Count(Test)
Business	0.5442	0.7025	0.6133	2678
Society	0.6493	0.5900	0.6182	1522
Arts	0.4838	0.5939	0.5332	1203
Shopping	0.4770	0.3811	0.4237	1004
Recreation	0.5392	0.5024	0.5201	847
Computers	0.5179	0.4564	0.4852	825
Sports	0.6501	0.6250	0.6373	648
Science	0.5229	0.3131	0.3917	511
Health	0.6503	0.4786	0.5514	442
Reference	0.5353	0.4631	0.4966	392
Games	0.5840	0.3967	0.4725	184
Home	0.6444	0.2320	0.3412	125
News	0.4821	0.3699	0.4186	72

Table 4 is summarized from the test prediction result of CNN-frozen. It presents how many fractions of domain names in a category are correctly/incorrectly assigned into another category. For instance, 62 "Arts" domain names are correctly assigned into the "Arts" category, while 13.1 are mistakenly labelled as "Business". The sum of each row is 100.

The result shows that "Business" is the most "popular" category in the prediction results. This is the main reason why its recall is the highest in Table 4. The "business" domain names that are correctly classified are likehttp://www.indianahealthinsurance.com/,http://www.timebussystems.com/,http://www.horsetrailerworld.com/, andhttp://www.insigniafutures.com/. On the other hands, a great number of domains names from other categories are mistakenly assigned to "Business". There are two reasons: 1) As discussed above, "Business" is the largest category in the training data. Thus, the classifier tends to label more "Business" domain names, especially for those with low confidence. 2) Conceptually, "Business" is highly broad: Almost everything can be a business. The boundary between business and other categories is ambiguous in nature. For example, a "Computers" domain namehttp://www.k12financials.com/, a website providing enterprise management solutions for K-12, is misclassified as "Business". Probably, the token "financials" misleads the classifier. However, the website is, in fact, relevant to business because it provides business services. Likewise, a "Shopping" domain namehttp://www.littlegreenworkshop.co.uk/ links to a website selling hand-crafted products. Its true cateogry is "Shopping", which has similar meaning with "Business". More accurately speaking, "Shopping" can be a hyponym of "Busienss". Thus, although it is misclassified into "Business", such a mistake can be tolerable in domain recommendation application: Recommendation candidates from "Business" (e.g.http://www.jeterphoto.com//) should also meet the needs of a user whose is looking for a "Shopping" domain name. By carefully design their own categories, registrars can avoid such misclassification.

The "Home" category is one of the smallest categories. Example domains in this category includesydneycitybonsai.org.au,vegetariansrecipes.org, andsocalplumeriasociety.com. The "Home" category has the least recall, meaning many domain names with true labels "Home" are misclassified into other categories. For example,http://www.fixya.com/ is a community based troubleshooting resource that provides consumer-generated, practical product tips. Consumers can get answers on products, such as appliances, office supplies, and cars. Since computers are covered in the website, misclassify it into "Computer" may not be an inexcusable error. Also,http://www.creditcardinsider.com/ is a website that provides suggestions on how credit cards work and how to build credit. It is not a business website that people can buy and sell goods or services. However, as the domain contains "creditcard", it is mistakenly signed to the "business" category.

6. Conclusions

The domain name service is a billion-dollar industry. In order to suggest relevant and available alternatives to registrants, it is important to understand the topic of the preferred domain name initially submitted.

The proposed method can be used as the first step of domain name recommendation. The classifier identifies the user intent by assigning the user-submitted domain names into one of the pre-defined topical

Table 4
Prediction Distribution of Individual Classes (Percentage).

	Predicted Category												
True Category	Arts	Business	Society	Shopping	Recreation	Computers	Sports	Science	Reference	Health	Games	Home	News
Arts	**62.0**	13.1	7.3	5.3	2.5	2.4	1.8	1.4	1.2	1.1	0.9	0.6	0.3
Business	7.7	**64.6**	5.8	7.2	2.7	4.2	1.5	2.5	0.9	2.3	0.4	0.2	0.1
Society	9.6	13.3	**60.2**	2.6	3.2	2.0	2.0	1.7	2.4	1.8	0.5	0.3	0.3
Shopping	16.0	22.3	5.7	**38.6**	4.3	3.0	3.7	1.3	0.7	2.8	1.0	0.5	0.1
Recreation	10.4	11.7	8.0	6.8	**51.6**	1.9	4.0	1.8	0.8	1.5	0.6	0.6	0.2
Computers	7.0	24.8	5.9	2.8	2.5	**47.2**	1.0	3.3	1.7	1.6	1.6	0.4	0.2
Sports	8.0	7.9	6.0	3.7	5.1	0.6	**64.8**	0.8	1.2	1.1	0.6	0.0	0.2
Science	6.3	24.3	8.2	2.5	2.7	6.7	1.2	**38.6**	4.5	3.5	0.8	0.4	0.4
Reference	6.9	14.8	15.3	0.5	1.5	3.3	2.0	6.4	**44.6**	1.5	0.5	1.0	1.5
Health	6.1	14.5	10.4	2.5	1.6	3.4	2.3	3.2	1.6	**54.1**	0.0	0.5	0.0
Games	13.6	11.4	12.0	3.8	4.3	0.0	2.7	1.6	1.6	0.0	**37.5**	0.0	0.0
Home	8.0	24.0	12.0	12.0	7.2	4.8	0.0	4.0	4.0	0.0	1.6	**22.4**	0.0
News	16.7	11.1	12.5	2.8	1.4	1.4	8.3	2.8	4.2	1.4	0.0	0.0	**37.5**

categories. The cost of the mis-classification can be significantly reduced by diversifying domain recommendations. Recommended domain names from different categories can be ranked by the predicted relevance. Top domains from diverse categories can be all shown on the first page. In this case, even if the most possible category is incorrect, users may still see relevant suggestions in the rest of the return list. The proposed method can improve user intent identification so that the correct sets of modifiers can be later triggered. The outcome can reduce users' effort of searching for relevant and available domain names. Also, it may boost domain registrars' revenue.

This paper addresses the problem of domain name classification, which is a critical part of domain name understanding. We propose to utilize subword information, and design deep neural network-based models to assign a domain name to one of 13 topical categories. We also propose three strategies to compute subword representation. Experimental evaluation verifies the effectiveness of our proposed models. The CNN with frozen subword embeddings learned from Wikipedia performs the best.

CRediT authorship contribution statement

Chong Wang: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft. **Yi Chen:** Writing - review & editing, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Lodico, P., 2017. 3 keys to the right domain name for building a brand. URL:https://www.entrepreneur.com/article/295947 (accessed 2018-06-19).

Styler, J., 2015. 5 tried-and-true tips for buying and selling domain names for profit. URL:https://www.godaddy.com/garage/5-tried-and-true-tips-for-buying-and-selling-domain-names-for-profit/ (accessed 2018-06-19).

Verisign, 2018. The verisign domain name industry brief (q1 2018). URL:https://www.verisign.com/en_US/domain-names/dnib/index.xhtml (accessed 2018-06-19).

ANI, 2019. The verisign domain name industry brief. URL:https://www.verisign.com/en_US/domain-names/dnib/index.xhtml.

Dunn, A., 2017. The world is running out of domain names-what will we do when they're all gone? URL:https://qz.com/994698/domain-name-registration-problems-are-going-to-become-a-lot-worse-when-we-run-out-of-names/.

Allemann, A., 2017. Breaking: Verisign loses appeal in.xyz lawsuit, URL:https://domainnamewire.com/2017/02/08/breaking-verisign-loses-appeal-xyz-lawsuit/.

Dunn, A., 2017. The world is running out of domain names-what will we do when they're all gone? URL:https://qz.com/994698/domain-name-registration-problems-are-going-to-become-a-lot-worse-when-we-run-out-of-names/ (accessed 2018-06-20).

Mei, J., Kou, X., Yao, Z., Rau-Chaplin, A., Islam, A., Moh'd, A., Milios, E.E., 2015. Efficient computation of co-occurrence based word relatedness. In: Proceedings of the 2015 ACM Symposium on Document Engineering. ACM, pp. 43–46.

Levy, O., Goldberg, Y., Dagan, I., 2015. Improving distributional similarity with lessons learned from word embeddings. Trans. Assoc. Comput. Linguistics 3, 211–225.

Asr, F.T., Zinkov, R., Jones, M., 2018. Querying word embeddings for similarity and relatedness. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. vol. 1 (Long Papers). pp. 675–684.

Tang, D., Qin, B., Liu, T., 2015. Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1432.

Zhou, L., Zhang, D., Yang, C.C., Wang, Y., 2018. Harnessing social media for health information management. Electron. Commerce Res. Appl. 27, 139–151.

Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 1245–1254.

Liu, H., Pan, X., Qu, Z., 2016. Learning based malicious web sites detection using suspicious urls. In: Proc. of the 34th International Conference on Software Engineering.

Le, A., Markopoulou, A., Faloutsos, M., 2011. Phishdef: Url names say it all. In: INFOCOM, 2011 Proceedings IEEE, IEEE. pp. 191–195.

Huang, D., Xu, K., Pei, J., 2014. Malicious url detection by dynamically mining patterns without pre-defined elements. World Wide Web 17 (6), 1375–1394.

Saxe, J., Berlin, K., 2017. Expose: a character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. arXiv:1702.08568.

Baykan, E., Henzinger, M., Marian, L., Weber, I., 2011. A comprehensive study of features and algorithms for url-based topic classification. ACM Trans. Web 5 (3), 15.

Abdallah, T.A., de La Iglesia, B., 2014. Url-based web page classification: with n-gram language models. In: International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Springer, Cham. pp. 19–33.

Rajalakshmi, R., Aravindan, C., 2018. A naive bayes approach for url classification with supervised feature selection and rejection framework. Comput. Intell. 34 (1), 363–396.

Hernández, I., Rivero, C.R., Ruiz, D., Corchuelo, R., 2014. Cala: an unsupervised url-based web page classification system. Knowl.-Based Syst. 57, 168–180.

Kim, Y., 2014. Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751.

Dos Santos, C., Gatti, M., 2014. Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 69–78.

Johnson, R., Zhang, T. Effective use of word order for text categorization with convolutional neural networks. arXiv:1412.1058.

Santos, C.D., Zadrozny, B., 2014. Learning character-level representations for part-of-speech tagging. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp. 1818–1826.

Severyn, A., Moschitti, A., 2015. Twitter sentiment analysis with deep convolutional neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 959–962.

Tang, D., Wei, F., Qin, B., Yang, N., Liu, T., Zhou, M., 2015. Sentiment embeddings with applications to sentiment analysis. IEEE Trans. Knowl. Data Eng. 28 (2), 496–509.

Zhang, X., Zhao, J., LeCun, Y., 2015. Character-level convolutional networks for text classification. Adv. Neural Inf. Process. Syst. 649–657.

Zhang, X., LeCun, Y. Text understanding from scratch. arXiv:1502.01710.

Ma, M., Huang, L., Zhou, B., Xiang, B., 2015. Dependency-based convolutional neural networks for sentence embedding. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 174–179.

Wang, X., Jiang, W., Luo, Z., 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 2428–2437.

Kim, Y., Jernite, Y., Sontag, D., Rush, A.M., 2016. Character-aware neural language models. In: Thirtieth AAAI Conference on Artificial Intelligence.

Liang, D., Xu, W., Zhao, Y., 2017. Combining word-level and character-level representations for relation classification of informal text. In: Proceedings of the 2nd Workshop on Representation Learning for NLP, Association for Computational Linguistics, Vancouver, Canada, pp. 43–47. https://doi.org/10.18653/v1/W17-2606.. URL:https://www.aclweb.org/anthology/W17-2606.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Lstrok, Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).https://doi.org/10.18653/v1/n18-1202.

Dai, W., Yu, Y., Zhang, C., Han, J., Xue, G.-R., 2006. A novel web page categorization algorithm based on block propagation using query-log information. In: WAIM. vol. 6. Springer. pp. 435–446.

Wang, W., Feng, Y., Dai, W., 2018. Topic analysis of online reviews for two competitive products using latent dirichlet allocation. Electron. Commerce Res. Appl. 29, 142–156.

Watrous-deVersterre, L., Wang, C., Song, M., 2012. Concept chaining utilizing meronyms in text characterization. In: Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries. ACM, pp. 241–248.

Kenter, T., Borisov, A., de Rijke, M. Siamese cbow: optimizing word embeddings for sentence representations. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.

Yu, L., Hermann, K.M., Blunsom, P., Pulman, S., 2014. Deep learning for answer sentence selection. NIPS 2014 Deep Learning and Representation Learning Workshop.

Gershman, S., Tenenbaum, J.B., 2015. Phrase similarity in humans and machines. In: Proceedings of the 37th Annual Conference of the Cognitive Science Society Citeseer.

Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. Bag of tricks for efficient text classification. arXiv:1607.01759.

Mikolov, T., Chen, K., Corrado, G., Dean, J. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning, pp. 1139–1147.

Foundation, W., 2003. Wikimedia. URL:https://dumps.wikimedia.org (accessed: 2018-11-20).

Jenks, G., 2018. Python word segmentation. URL:https://pypi.org/project/wordsegment/ (accessed: 2018-11-19).

Norvig, P., 2009. Natural language corpus data. Beautiful Data 219–242.

Brants, T., Franz, A. Web 1t 5-gram version 1.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.