

A Machine-Learning Approach to Discovering Company Home Pages

Wojciech Gryc
Oxford Internet Institute
University of Oxford
Oxford, UK OX1 3JS
Email: wojciech.gryc@oii.ox.ac.uk

Prem Melville
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
Email: pmelvil@us.ibm.com

Richard D. Lawrence
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
Email: ricklawr@us.ibm.com

Abstract—For many marketing and business applications, it is necessary to know the home page of a company specified only by its company name. If we require the home page for a small number of big companies, this task is readily accomplished via use of Internet search engines or access to domain registration lists. However, if the entities of interest are small companies, these approaches can lead to mismatches, particularly if a specified company lacks a home page. We address this problem using a supervised machine-learning approach in which we train a binary classification model. We classify potential website matches for each company name based on a set of explanatory features extracted from the content on each candidate website. Our approach is related to web-based business intelligence in two ways: (1) we build the training set for our learning algorithms through crowdsourcing tools and illustrate their potential for business research, and (2) the success of our model allows one to easily use corporate home pages as data inputs into other research projects. Through the successful use of crowdsourcing, our approach is able to identify a correct home page or recognize that a valid home page does not exist with an accuracy that is 57% better than simply taking the highest ranked search engine result as the correct match.

I. INTRODUCTION

For many business-related applications, it is useful to know the Internet home page (or URL) for a specified set of companies. If the companies are large, e.g. Fortune 500 companies, this task can be accomplished easily by submitting each company name to an Internet search engine and capturing the first returned result. However, if we require the home page for a very large number of smaller companies, then such an approach is no longer feasible due to the sheer number of companies, and the observation that the first search return for a smaller company is less likely to be the correct home page. Indeed, many small companies will have no website at all, so it is important that an automated capability detect such cases.

There are several major data vendors that offer detailed “firmographic” information on a very large number (order millions) of companies, including very small companies. These data include information such as annual revenue, number of employees, the primary industry in which the company operates, and contact information including the company website.

We acknowledge Ildar Khabibrakhmanov for his contributions during the initial phase of this work.

As noted above, it cannot be assumed that essentially all companies listed in such directories have a website. Specifically, in a random sample of 947 companies with annual revenue between \$3M and \$100M, 372 (39%) of these companies did not have an observed website. Hence, correct resolution of a company’s website, including those without a site, is a key data quality issue for companies that sell such data.

An important application that requires reliable website identification arises when we seek to join structured firmographic information for a large number of small companies with indexed content crawled from their respective websites. The resulting database of merged structured and unstructured content provides a rich set of data for machine-learning applications [5], and can also enable new focused search capabilities to address a range of marketing objectives. A key initial step in this process is automatically determining the correct home page for the tens of thousands of companies required in this application.

To be able to harness information from company websites, one first needs an automated way to know when one is viewing the home page of a given company. Through the use of crowdsourcing, we build a large training set and illustrate that the data obtained through this method is reliable. Using this data, we develop a machine-learning approach to discovering the home page for a specified company, and compare results with the non-learning approach of accepting the first return from an Internet search engine.

II. PROBLEM SPECIFICATION

As noted in the introduction, our objective is to determine the correct home page for a company specified only by its name. We refer to this overall objective as the *Mapping Task*, where a specific URL is *mapped* onto a company name. This task can be formulated as a supervised, machine-learning problem by generating a set of potential home page (URL) matches (or candidates) for each company, and then training a binary classification model against a labeled set of such examples. Such a model estimates the probability that a specific URL represents the home page of a given company. We refer to the process of scoring each individual candidate for a company name (a company name and URL match) as the *Match-Evaluation Task*, or simply as the *Matching Task*.

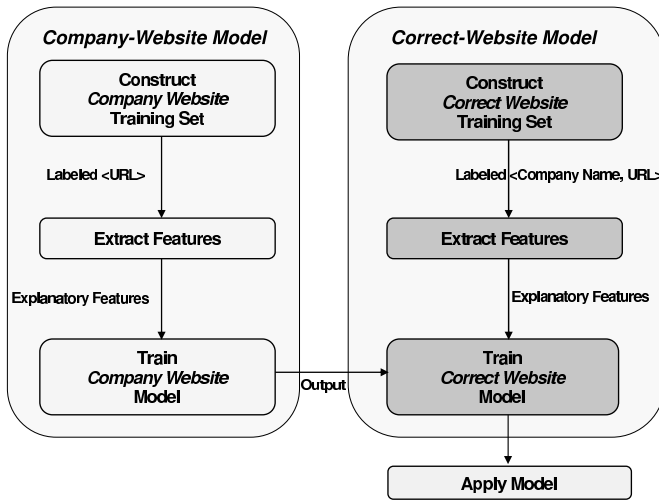


Fig. 1. Training the Matching Model

The *Mapping Task* then aggregates these individual results to obtain a final recommendation for the specified company name.

The generation of candidate URLs is accomplished by submitting the company name to an Internet search engine, and retaining a small number of unique domains returned by such a query as candidate URLs for this company name. Explanatory features are obtained via analysis of company name and the HTML tags and content on the candidate URL. Using these features, each candidate match is scored by a model (denoted as the *Correct-Website Model*) trained to estimate the probability that a given candidate match is correct. The best match is returned as the final result of the mapping task. If none of the candidate matches meets a specified confidence, the company is assumed to have no website.

Figure 1 illustrates the training of the *Correct-Website Model*. Note that this model accepts output from a second model designed to estimate the probability that a given URL is the home page of *any* company. We refer to this as the *Company Website Model* – it is also implemented as a binary classifier.

The objective of each model can be made clear via the different training sets used in each. Given a Company i and candidate URL j , we denote a candidate match as $\langle \text{CompanyName}, \text{URL} \rangle$. For the *Correct-Website Model*, we label each such match as correct or incorrect based on visual inspection of the website under the following definition:

A candidate match $\langle \text{CompanyName}, \text{URL} \rangle$ is a correct example if it appears likely that this site was created by the named entity, either directly, or indirectly via a third-party website developer. The match is also considered correct if it is clear that the company name is a subsidiary or recent acquisition of the parent company identified by the *URL*.

The training set for the *Company Website Model* is constructed by manually labeling a set $\{\text{URL}_j\}$ as positive or negative based on whether the site appears to be the home page for an

actual company. This can be somewhat subjective, and we use the following guidance in generating these labels:

A URL_j is a positive example if it clearly belongs to a business entity that produces some kind of goods or services, and the site appears to have the goal to promote some aspect of their business model. We also label not-for-profit organizations like foundations and hospitals as positive examples for this model.

The following sections elaborate on the specifics of the models as outlined above.

III. DATA SETS

In this section, we describe the approach used to obtain the labeled data sets mentioned in the previous section. We choose a set of company names listed by Dun & Bradstreet (<http://www.dnb.com>), which maintains information on over 15 million companies worldwide. We restricted ourselves to companies in the US with an annual revenue greater than \$3 million, which yields a set of approximately 375,000 companies. This set is then sampled uniformly to yield 1087 companies.

For each company name in this base list, we need to generate a set of potential URL matches or candidates. We build the list of candidates by submitting the company name to Google. Before submitting each company name, any legal identifiers (e.g. “Inc” or “LLC”) are stripped from the company name, and the remaining text is encapsulated in quotation marks to ensure that sites with the full company name appear in the search results.

Depending on the uniqueness of the search terms, the number of results for each submitted company name can vary from zero (in a small number of cases) to hundreds of thousands. For our analysis, we retain only the top ten URLs as ordered by Google. Since the goal of this task is creating candidate home page matches for each company, the resulting sites are stripped of any subdirectories to focus specifically on the domain. Each unique domain name here becomes a candidate match for the specified company. Note that due to the limited numbers of search results, some companies will have fewer than ten candidates.

As a result of this process, we have a set of potential $\langle \text{CompanyName}, \text{URL} \rangle$ matches. We use this list to generate the set of training examples through labeling by the authors, as well as using the “Amazon Mechanical Turk” service, described later in the paper. This process results in the following training sets:

- For 103 companies within the list of 1087, we label the set of unique URL_j to form the training set for the *Company Website* model, labeled as positive or negative using the criteria given in Section II. This yields a labeled set of 628 examples, 176 of which are positive.
- The set of unique $\langle \text{CompanyName}, \text{URL} \rangle$ pairs form the training set for the *Correct Website* model, labeled as positive or negative. This yields a labeled set of 8551

examples, 802 of which are positive. Note that some companies have more than one positive result due to the use of multiple domain names or having subsidiary-specific home pages as well.

To aid in labeling, matches were submitted to the Amazon Mechanical Turk service (<http://www.mturk.com/>). This service allows one to submit a simple question, called a ‘‘Human Intelligence Task’’ (HIT), and have an individual (or ‘‘Worker’’) provide an answer to each HIT in exchange for a nominal reward. For our labeling process, 9703 pairs were submitted as HITs, representing a total of 984 unique companies.

To ensure a high quality of labeling, each $\langle CompanyName, URL \rangle$ pair was labeled as correct or incorrect by 3 different workers. Validation by the authors showed that mistaken labeling tends to occur through false positives (i.e. a match is incorrectly labeled as correct) rather than false negatives. To deal with this, the 984 companies were filtered to include only the companies for which the authors are confident in their labels. A candidate $\langle CompanyName, URL \rangle$ match was accepted as correct only if all 3 workers labeled it as correct. A company name was assumed to have no URL if none of the 10 candidate matches for this company received more than 1 correct label from the 3 workers. If neither of these conditions were met for a specific company name, then the result was viewed as uncertain, and the company was discarded from further analysis. In this way, the number of companies labeled using the Mechanical Turk service was reduced from 984 to 844 companies.

A major concern in terms of building the training set was the quality of data collected on the Mechanical Turk service. Each $\langle CompanyName, URL \rangle$ pair was labeled by three independent workers, allowing one to estimate the proportion of times that the pairs will receive unanimous votes given a specific worker participating in that labeling task. This provides a heuristic for estimates worker quality. Figure 2 shows the distribution of proportions (i.e. estimated probability) for the 389 workers. The histogram shows that the data can generally be trusted, as the mean value for workers is 0.77. If we take these proportions as probability estimates for workers being ‘‘correct’’, then we are above the 0.5 threshold set by other researchers for data quality [7].

Combined with the examples labeled by authors, this resulted in a data set consisting of 947 companies, and 8551 $\langle CompanyName, URL \rangle$ pairs.

IV. FEATURE CONSTRUCTION

We separate these features into two sets:

- structured features that reflect analysis of a website’s HTML tags and body, as well as other structural properties of the page such as hyperlinks; and,
- unstructured text features extracted from the raw content that appears on the web page.

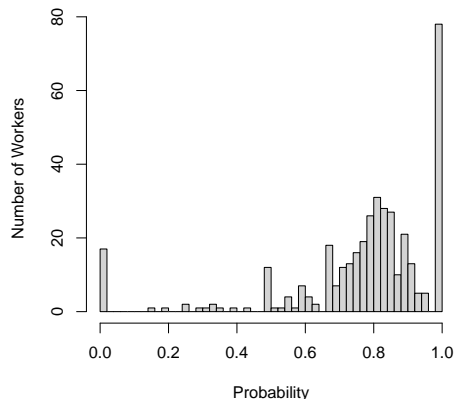


Fig. 2. Histogram of the quality of Mechanical Turk workers. Estimated probabilities near 1.0 show workers who have high quality labels.

A. Structured features

Given a $\langle CompanyName, URL \rangle$ candidate match, the structured features capture the extent to which the company name is present in either the HTML tags or the body of the page at this URL. The structured features also capture other non-content-based attributes of the page, such as the number of ads on the page. For the sake of brevity, an in-depth analysis of structured features is not included in the paper, but is available upon request.

The general structure of a web page includes header information, encapsulated in a $\langle head \rangle$ tag, followed by the actual web page itself (as shown in a web browser) surrounded by a $\langle body \rangle$ tag. The title of the page itself is contained within a separate $\langle title \rangle$ tag embedded in the header. Many of the features capture analysis of the $\langle title \rangle$ tag, effectively measuring the extent to which the company name is reflected in the title. The Levenshtein distance [3] is one such measure of edit distance, effectively measuring the number of operations required to transform one string into another.

Another strategy we employ is to examine the hyperlink information and the link structure of a web page. Examples of the resulting features include the number of links to domains known to host ads and the number of links to other websites. These features are intended to capture the observation that directories or portals often contain a large number of links and very little content, and hence can be excluded as potential matches for a specific company. However, aside from counting ad-related domains, it appears that the hyperlink structure provides little useful information for the model, as measured by the information gain.

If external information about a candidate website is available, this can prove extremely valuable in feature creation as well; where, by *external* we refer to features that cannot be extracted from the content of the website itself. Since candidate matches include a URL obtained by submitting the company name to Google, we have both the domain name of

the URL and the rank of this URL in the ordered list of sites returned by the Google search.

B. Unstructured text features

Apart from the structured features described in the previous section, we can also exploit the unstructured text content that appears on each candidate website. The web content by itself can be quite useful in determining if a candidate web page is a valid company site, as opposed to the website of a person, directory, web log, etc. For example, words like advertisement, yellow, directories, citysearch appear on web pages of directory services that might list company names, but are not the home page of the desired company. On the other hand, words such as products, services, contact are usually more commonly found on actual company websites. If a candidate URL is not the home page of *any* company, then we can be sure that it is not a correct match for the specific company in question. One could build a list of keywords by hand to help determine if a page is likely to be a company page or not. However, we find it more efficient and accurate to train models to automatically learn this distinction. We describe this process in more detail below.

For each unique candidate URL in our data, we download the corresponding web page and pre-process the text by removing stop words, stemming the words into inflected forms, and filtering out words that appear in less than 25 web pages. This results in a collection of 5777 web pages, represented by 4818 unique words, which we convert into vectors using the bag-of-word representation with TF-IDF term weighting [1]. Note that there are less web pages than total $\langle CompanyName, URL \rangle$ pairs because some URLs, like those of business directories, may be candidates for multiple companies.

C. Analyzing text features

Given the data described above, we can now build text-based models to predict if a given URL is the web page of any company. We use the *Company Website* labels of the 103 company subset described in Section III. Since this is a smaller data set, less stringent pruning is applied to the text data, with words discarded if they appear on less than five web pages. We then train a naïve Bayes classifier using a multinomial text model [4]. As noted by Rennie et al. [6] and Frank and Bouckaert [2] naïve Bayes trained on imbalanced data produces predictions that are biased in favor of large classes. To overcome this, we re-weighted the instances in the training data so that a positive instance has 3.5 times the weight of a negative instance (which corresponds to the imbalance in this data set). We evaluated this classifier using 10-fold cross-validation and present the resulting ROC curve in Figure 3. The results demonstrate that our model is quite effective in separating company pages from non-company pages, based solely on the text that appears on the page — producing a model AUC of 0.809.

Since we are primarily interested in finding correct URL matches for the company names in our data, we can further refine our *Company Website* model to only focus on the

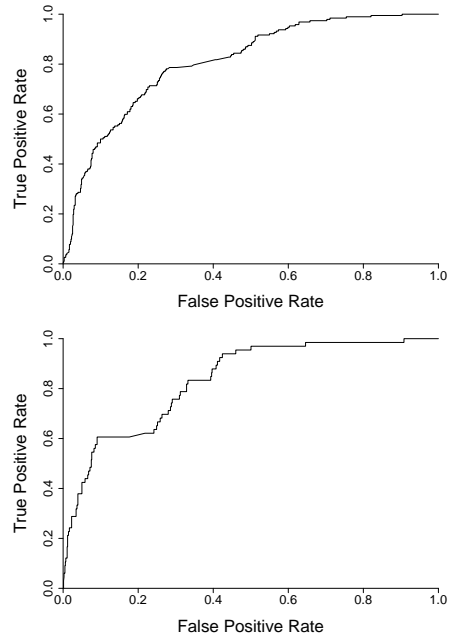


Fig. 3. ROC curve for text-classification model trained to identify company websites (top), and for a focused *Company Website* model trained to identify websites for companies drawn from the distribution of correct matches (bottom).

distribution of companies in our dataset. We do this by training our text classification models using only the company websites of correct matches in our dataset as positive examples of companies (as opposed to all company websites). This focused *Company Website* model performs even better, resulting in an AUC of 0.833 for the ROC curve shown in Figure 3. These results confirm that the raw text content is indeed quite a useful source of information for building *Company Website* models, which in turn can be used to drive accurate *Correct Website* models.

V. THE URL-MATCHING MODELS

In this section, we describe different approaches to combining the structured and unstructured features to build more accurate *Correct Website* models. We compare the following three approaches of incorporating the *Company Website* model into a *Correct Website* model.

Voting: In this approach, we train a separate naïve Bayes *Company Website* model and a logistic regression *Correct Website* model (on the same training instances), and then average the class probability estimates produced by both models to get a revised estimate of a correct match.

Nesting: In this approach, we use the output of the *Company Website* model as an input to the logistic regression model. Specifically, we add another variable in the logistic regression, corresponding to the predicted probability that the candidate website is a company website as given by a *Company Website* model. In order not to bias our evaluation, we build the *Company Website* model using the same training set as used by the logistic regression model. However, in order

to train the logistic regression on the additional *Company Website* score, we need to provide values for this variable on the training data. We could do this by training a *Company Website* model on the training set and providing the scores on the same training data. However, the logistic regression trained on this input could be prone to over-fitting. Hence, a better approach is to use cross-validation on the training set to get unbiased scores from a *Company Website* model, i.e., the training set is further split into 10 folds, and the instances in each fold are scored by a *Company Website* model that has been trained on the remaining 9 folds. These unbiased estimates are then used as inputs to the logistic regression along with all the other structured features. In experiments, not presented here, we have confirmed that this unbiased estimate approach does in fact improve on using training set scores.

Stacking: This approach is similar to voting, where two separate models are trained on the structured and unstructured features using a naïve Bayes classifier and logistic regression respectively. However, instead of simply averaging the output predictions, we train another logistic regression model to learn how to optimally combine the outputs of the base models — following Wolpert’s [8] approach to Stacked Generalization.

VI. EXPERIMENTAL EVALUATION

A. Evaluating the matching task

We evaluated the three *Correct Website* models described in Section V on our set of 8551 $\langle \text{Company Name}, \text{URL} \rangle$ pairs. Since candidate URLs are retrieved through a search engine, a natural approach to URL selection is to simply pick the highest ranked result. We refer to this baseline approach as the “I’m Feeling Lucky” (IFL) classifier. It classifies all first search returns as a correct match, and all other candidate URLs are assumed to be incorrect matches. We also present results for the *Correct Website* models built using only the structured features described in Section IV-A.

All experiments were performed using 10-fold cross-validation, and ROC curves are shown in Figure 4. For clarity of the figure, we only plot the three most relevant curves. Note that the IFL classifier only outputs predicted class labels, and does not provide class membership probabilities. As such, it only presents the single operating point of the IFL classifier. In lieu of an ROC curve, we present the confusion matrix for the IFL classifier in Table I.

Of the 947 companies in the data set, 575 have home pages whose domains appear in the search results. Of these 575 companies, 474 have their home pages as the top rank in the set of results. Thus, it is clear that the top ranking sites are often the correct ones, if one assumes a home page exists. However, only 60.7% of company names yield a site, making the IFL classifier generally inaccurate. IFL classification yields a matching accuracy of 90.6%; which may seem high, but in fact is equal to the base rate, i.e. classifying all examples as negative.

All the modeling-based approaches perform substantially better when compared to the baseline IFL classifier. This is evidenced by the fact that even the single operating point

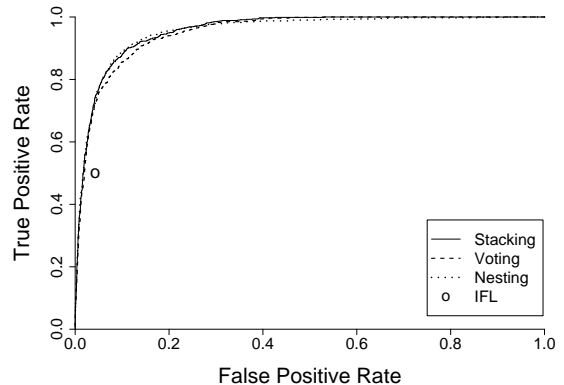


Fig. 4. ROC curves for the three URL-matching models compared to the baseline “I’m Feeling Lucky” performance.

TABLE I
CONFUSION MATRIX FOR THE IFL CLASSIFIER.

| | | Predicted | |
|--------|----------|-----------|----------|
| | | A Match | No Match |
| Actual | A Match | 474 | 328 |
| | No Match | 473 | 7276 |

of IFL lies below the ROC curves of our matching models in Figure 4. The results also confirm that incorporating the unstructured text features via the *Company Website* model can improve on using only structured features. Amongst the three models that use both structured and unstructured features, Stacking shows the best performance in terms of AUC. Clearly, learning how to combine the outputs of the *Company Website* model with the logistic regression model on structured features, as done by Stacking, is a very effective approach to solving the URL-matching problem. As such, we use this approach for the evaluation in the following section.

B. Evaluating the mapping task

The experimental results thus far show that evaluating a $\langle \text{Company Name}, \text{URL} \rangle$ match can be accomplished with an AUC of over 0.95. However, as stated in Section II, this matching problem is a sub-task of the real problem of URL-mapping — i.e., given a company name, map it to the appropriate home page URL. Given a *Correct Website* classifier, the mapping task can be solved in the following way.

For a specific company name, a set of N (in our case, $N = 10$) candidate URLs is obtained from a search engine, resulting in $N \langle \text{Company Name}, \text{URL} \rangle$ pairs. Each pair is then passed on to the *Correct Website* model, which returns a probability representing the likelihood that the pair is a correct match. If at least one probability score is above a specified threshold value, T , then the highest-ranked pair is considered to be the correct match, and the company name is then mapped to that URL. Otherwise, we assume that all candidate URLs are incorrect and that the company does not have a home page.

The threshold, T , can be selected based on the relative costs or penalties of having false-positive versus false-negative

TABLE II
COMPARISON OF THE IFL CLASSIFIER VERSUS MODEL-DRIVEN URL MAPPING USING TWO DIFFERENT THRESHOLDS.

| Company Type | # Companies | Predicted Class | IFL | T = 0.17 | T = 0.90 |
|--------------------|-------------|------------------------|-------|----------|----------|
| With Valid Site | 575 | Correct | 474 | 445 | 129 |
| | | Incorrect (Missing) | 0 | 47 | 431 |
| | | Incorrect (Wrong Site) | 101 | 83 | 15 |
| Without Valid Site | 372 | Correct | 0 | 298 | 370 |
| | | Incorrect (Wrong Site) | 372 | 74 | 2 |
| All | 947 | Accuracy | 50.1% | 78.5% | 71.8% |

matches. We consider two such cases, where $T = 0.17$ and $T = 0.90$. Both results, along with the results of the “*I’m Feeling Lucky*” classifier, are presented in Table II. In addition to overall accuracies, we also report results separately for companies that have a valid website, and those that do not.

For companies without a website, two outcomes are possible: classifying all candidates as incorrect, or incorrectly identifying one candidate as the company home page. In the case of the IFL classifier, the highest ranked result is always considered to be the correct one, and so it incorrectly identifies home pages for all companies without sites. Our model, on the other hand, is able to identify 298 (80.1%) and 370 (99.5%) companies as not having home pages, for $T = 0.17$ and $T = 0.90$ respectively.

For the 575 companies that do have a home page, there are three possible outcomes: (1) the correct home page is mapped to the company, (2) the company is incorrectly classified as not having a home page, and (3) an incorrect home page is mapped to the company name. For this specific subset of companies, the IFL classifier does best, mapping 474 (82.4%) companies to their home pages, while our model does so for 445 (77.4%) and 129 (22.4%) companies at $T = 0.17$ and $T = 0.90$ respectively. However, when the IFL classifier does not predict the correct home page, it maps the company to an incorrect page. Our model tends to label companies as not having sites rather than providing an incorrect home page when it is uncertain of the right match. This can be seen in Table II, where our models for $T = 0.17$ and $T = 0.90$ provide an incorrect home page for only 83 (14.4%) and 15 (2.6%) of companies respectively, which is less than the IFL classifier. While our model (at the specified thresholds) has higher levels of predicting websites as missing, it is arguable from a data-quality standpoint that this is much better than having incorrect websites.

The overall accuracies of our mapping algorithm are quite high relative to the IFL baseline, which has an accuracy of only 50.1% for all 947 companies compared to our models’ 78.5% and 71.8%, respectively for thresholds of 0.17 and 0.90. Clearly, selecting the first return of a search engine is not a viable solution for automating the process of discovering company home pages. In contrast, our model-driven approach can perform over 56% better than this baseline, and provides the added ability of being able to reject all candidate URLs for a company that has no website. The IFL classifier can only do this in the trivial case, when a search engine returns zero results.

VII. CONCLUSION

Determining the home page for a company identified only by its name is not a trivial task because many small companies have a limited web presence. Based on our research, almost 40% of small companies do not have a website that appears in the top ten search engine results. In this paper, we describe how the task of mapping a company name to a home page can be solved through a sub-task of evaluating if a given candidate URL matches the company name. We formulate this sub-task of evaluating matches as a supervised learning problem, and solve it by training classifiers on features that analyze website structure as well as text content. The development of such a classifier is dependent on the use of crowdsourcing tools to build a large and robust training set for our algorithms. Using such a classifier based on a Stacked learner, we are able to achieve an accuracy of 78.5% in mapping a correct URL to a company name, with only 16.6% of company names mapped to an incorrect search result. A reasonable baseline method is to simply take the top-ranked search result as the company home page. Such an approach yields an accuracy of only 50.1%. Our approach has the added benefit of being able to accurately predict that there are no reliable candidate URLs for a company name.

REFERENCES

- [1] C. Buckley, G. Salton, and J. Allan, “The effect of adding relevance information in a relevance feedback environment,” in *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag, 1994, pp. 292–300.
- [2] E. Frank and R. R. Bouckaert, “Naïve Bayes for text classification with unbalanced classes,” in *Proceedings of 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2006, pp. 503–510.
- [3] V. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” *Soviet Physics Doklady*, vol. 10, p. 707, 1966.
- [4] A. McCallum and K. Nigam, “A comparison of event models for naive Bayes text classification,” in *Papers from the AAAI-98 Workshop on Text Categorization*, Madison, WI, Jul. 1998, pp. 41–48.
- [5] P. Melville, S. Rosset, and R. Lawrence, “Customer targeting models using actively-selected web content,” *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 946–953, 2008.
- [6] J. Rennie, L. Shih, J. Teevan, and D. Karger, “Tackling the poor assumptions of naive bayes text classifiers,” in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003, pp. 616–623.
- [7] V. Sheng, F. Provost, and P. Ipeirotis, “Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 614–622.
- [8] D. H. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, pp. 241–259, 1992.