Vertical Selection in the Presence of Unlabeled Verticals

Jaime Arguello[*]
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA, USA
jaime@cs.cmu.edu

Fernando Diaz
Yahoo! Labs
4301 Great America Pkwy
Santa Clara, CA, USA
diazf@yahoo-inc.com

Jean-François Paiement
Yahoo! Labs
1000 Rue de la Gauchetiere
Suite 2400
Montreal, QC, Canada
paiement@yahoo-inc.com

ABSTRACT

Vertical aggregation is the task of incorporating results from specialized search engines or verticals (e.g., images, video, news) into Web search results. Vertical selection is the subtask of deciding, given a query, which verticals, if any, are relevant. State of the art approaches use machine learned models to predict which verticals are relevant to a query. When trained using a large set of labeled data, a machine learned vertical selection model outperforms baselines which require no training data. Unfortunately, whenever a new vertical is introduced, a costly new set of editorial data must be gathered. In this paper, we propose methods for reusing training data from a set of existing (source) verticals to learn a predictive model for a new (target) vertical. We study methods for learning robust, portable, and adaptive cross-vertical models. Experiments show the need to focus on different types of features when maximizing *portability* (the ability for a single model to make accurate predictions across *multiple verticals*) than when maximizing *adaptability* (the ability for a single model to make accurate predictions for a *specific vertical*). We demonstrate the efficacy of our methods through extensive experimentation for 11 verticals.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

vertical search, distributed information retrieval, query intent, domain adaptation

[*]work done while at Yahoo! Labs Montreal

1. INTRODUCTION

Many web search engines provide their users with access to specialized search services, or *verticals*, that focus on a specific type of media (e.g., blogs, images, video) or domain (e.g., health, music, travel). In some cases, if a user is aware of a relevant vertical, the query can be issued directly to a vertical-specific search engine. However, to improve user-experience, portal search engines have started embedding relevant vertical content in Web search results. This has been referred to as *aggregated search*. A necessary part of an aggregated search system is *vertical selection*, the task of deciding, given a query, which vertical back-ends, if any, should be presented alongside Web search results.

Vertical selection can be viewed as a type of resource selection, a well-studied task in distributed information retrieval. Recent work shows that a supervised machine learning approach to vertical selection outperforms traditional resource selection methods [2, 1]. Most of these traditional methods derive evidence exclusively from vertical content and do not require extensive training data [4, 15, 18, 17, 16, 19].

While a supervised approach outperforms state-of-the-art methods, one of its drawbacks is that it requires training data. As is shown in Arguello *et al.* [2], suitable training data (e.g., vertical relevance judgments for a set of queries) can originate from human annotations. Human annotation is resource intensive in terms of time and money. An annotation effort may be sensible as a one-time investment. However, in practice, the aggregated search environment is often dynamic. Verticals can be added to or removed from the set of candidate verticals. Documents can be added to or removed from a vertical. The interests of the user population may drift, in effect, changing the set of vertical documents most likely to be requested by users. It may not be feasible to annotate a new set of queries every time the environment undergoes a significant change.

In this paper, our goal is to improve a system's return on editorial investment. We are interested in the following scenario. Suppose we have a set of verticals for which we have collected training data in the form of human vertical relevance judgements. We refer to these verticals as the *source* verticals. Then, suppose we have a new vertical associated with no training data. We refer to this vertical as the *target* vertical. Our objective is to learn a predictive model for the target vertical using *only* source-vertical training data.

Our solution to this problem focuses on two model properties: *portability* and *adaptability*. A portable model is one that can be effectively applied to *any* target vertical with no additional training data. If a model is portable, it can be used as a 'black box' for any new vertical. An adaptable model is one that can be tailored to a specific new vertical with no additional training data. If a model is adaptable, its parameters can be automatically adjusted to suit the new vertical *at no editorial cost*. We will present models which exhibit these properties and evaluate their performance across a set of 11 target verticals.

2. RELATED WORK

In distributed IR, most traditional approaches to resource selection cast the problem as resource ranking and score collections using functions tuned manually on a few training queries. Most of these approaches focus exclusively on the similarity between the query and the collection content, possibly using sampled documents [4, 15, 18, 17, 16, 19]. Of these, so-called large document models treat each collection (or its document samples) as a single monolithic document and adapt document ranking functions to rank collections [4, 18]. In contrast, small document models focus on those (sampled) collection documents most relevant to the query [15, 17, 16, 19]. ReDDE [17], for example, scores collections by their expected number of relevant documents. This expectation is based on the number of collection samples predicted relevant, scaled by a factor proportional to the collection size.

Other approaches cast resource selection as a supervised machine learned classification [2, 1] or rank-learning task [20]. In contrast to the single-evidence resource-scoring methods described above, these approaches have the advantage of easily incorporating multiple types of evidence as input features, for example, resource-specific query-logs [2] or click-through data [1]. This type of evidence integration is critical when dealing with text-impoverished verticals (e.g., images, video), which are problematic for methods that focus exclusively on content-based evidence.

In machine learning, domain adaptation is the task of using training data from one or more *source* domains to learn a predictive model for a *target* domain, typically associated with little or no training data. While domain adaptation has not been studied in the context of supervised vertical selection, it has been widely studied in other applications. The domain adaptation problem arises from the fact that the source and target data originate from different distributions.

One line of work performs instance weighting to down-weight the influence of "misleading" source-domain training instances. Jiang and Zhai [11] do this by discarding source-domain training instances that are misclassified by a model trained on whatever target-domain training data is available. This is slightly different than our problem setting since we assume that we have no training data in the target domain.

A different approach is to perform feature weighing to learn a model that generalizes better to the target domain. Saptal and Sarawagi [14] perform feature subset selection by minimizing a distance function between a single source domain and target domain data distribution. The target data distribution is estimated using predicted labels from a source-trained model. This is different from our work since we want to learn from multiple source domains. Jiang [10] proposes an approach that uses training data from multiple domains. First, a generalizable subset of features is identified based on their predictiveness across source domains. Then, a model that focuses heavily on these features is used to produce predictions on the target domain. Finally, a target-domain classifier with access to all available features is trained on these predictions. This approach assumes binary features and requires using linear classifiers. The framework we propose can handle real-valued features and models that can explore complex features interactions.

Feature subset selection can be viewed as a type of representation change aimed to make a model more generalizable to the target domain. A similar technique is feature augmentation. Given access to some target-domain training data, Daumé III [6] augments the feature space by making three versions of each feature: a source-domain, target-domain, and general version. A model is then trained on the union of the source and target training data. This allows the model to effectively weight a source of evidence differently when it is predictive of the target class in only the source domain, only the target domain, or both. Blitzer *et al.* [3] propose an approach that does this without requiring target-domain training data. The goal is to identify the correspondence of *non-pivot* features, which may have a different correlation with the target class across domains, based on their predictiveness of *pivot* features, manually identified as similarly correlated with the target class across domains. The assumption is that the cross-domain correspondence between pairs of non-pivot features is encoded in the weights assigned to each when training linear classifiers to predict the presence of each pivot feature using non-pivot features.

A different approach is to train a model on the source domain and then to only fine-tune it using a few target-domain training examples. In the context of web search, Chen *et al.* [5] propose several ways to fine-tune a model trained using Gradient Boosting Decision Trees, which combines weak models into a more complex model. The approach of appending trees (based on target-domain training data) to a source-trained model has the advantage of accommodating features only present in the target domain.

3. PROBLEM FORMULATION

Let $y_v(q)$ denote the true relevance label of vertical v with respect to query q. In the general vertical selection setting, the goal is to learn a function f that approximates y. In this work, we focus on the following scenario. Assume we have a set, $\mathcal{S}$, of source verticals each with labeled queries. Then, suppose we are given a new (target) vertical t with no labeled data. Our objective is to learn a function f that approximates y_t using *only* source-vertical training data. The quality of an approximation will be measured by some metric that compares the predicted and true query labels. We use notation,

$$\mu(f, y_t, \mathcal{Q})$$

to refer to the evaluation of function f on query set $\mathcal{Q}$. This metric could be classification based (e.g. accuracy) or rank-based (e.g. average precision).

4. FEATURE-BASED MODEL

In our work, the domain of f is not the universe of possible query strings. Instead, we generate features of the query string which we believe correlate with vertical relevance and are generalizable across queries. In this section, we review the basic feature-based vertical selection model. A more detailed description of our features can be found in our references [2, 1].

4.1 Features

Given query q and vertical v, let $\phi_v(q)$ be a vector of features. Whatever the vertical, the semantics of a particular feature are the same. For example, if $\phi_v(q)_i$ refers to the number of times the query was issued by users directly to v, then $\phi_{v'}(q)_i$ refers to the number of times the query was issued to v'. As a result, all feature vectors are of the same length.

We generate two types of features,

1. *query-vertical features*: specific to the query-vertical pair, for example, the number of times the query was previously issued directly to the vertical by users.

2. *query features*: specific to the query and independent of the vertical, for example, whether the query is related to the *travel* domain.[1]

Our approach is to use signals shown to be useful for supervised vertical selection in previous work. We describe them briefly below.

4.1.1 Query-Vertical Features

Query-vertical features are generated from the vertical. In our case, they are derived from the similarity between the query and sampled vertical content and from the similarity between the query and queries issued previously to the vertical by users.[2] We use five query-vertical features.

The first four query-vertical features are generated using a retrieval from a centralized sample index, an index that combines documents sampled from every vertical. ReDDE.top [2, 1] is a variation of ReDDE [17]. GAVG measures the geometric average document score of the vertical's top sampled documents [15]. Soft.ReDDE [2] is a variation of ReDDE.top and uses a soft document-to-vertical membership. We generated two Soft.ReDDE features. One version uses a document-to-vertical membership based on the similarity between the document and a language model constructed from vertical-sampled documents. A second version uses the similarity between the document and a language model constructed from the vertical query-log. Finally, we use the query likelihood given the vertical's query-log language model. Each query-vertical feature was mass normalized across verticals.

4.1.2 Query Features

Query features are generated from the query, independent of the vertical. These features are used in previous work and are described more completely in Arguello *et al.* [2]. Boolean features include regular expressions and dictionary look-ups likely to correlate with vertical intent (e.g., does query contain the keyword "news"?). Geographic features correspond to the output of a geographic named-entity tagger (e.g., does the query contain a city name?). Categorical features correspond to the output of a query-domain categorization algorithm (e.g., is the query related to the travel domain?). In total, we use about 120 query features.

[1]We can imagine additionally having query-independent *vertical features*, for example, whether the vertical has observed a sudden increase in query traffic. We do not make use of vertical features.

[2]Vertical-specific query-logs are available to the system under the assumption of a cooperative environment.

4.2 Learning Algorithm

We adopt a machine learning approach to vertical selection. In all experiments, we used the Gradient Boosted Decision Trees (GBDT) algorithm [9]. The main component of a GBDT model is a regression tree. A regression tree is a simple binary tree. Each internal node corresponds to a feature and a splitting condition which partitions the data. Each terminal node corresponds to a response value, the predicted output value. GBDT combines regression trees in a boosting framework to form a more complex model. During training each additional regression tree is trained on the residuals of the current prediction. In our case, we chose to minimize logistic loss, which has demonstrated effective performance for vertical selection in prior work [1, 2, 8, 7]. That is, the model maximizes,

$$\mu_{\log}(f_v, y_v, \mathcal{Q}) = -\sum_{q \in \mathcal{Q}} \ell_{\log}(f_v(\phi_v(q)) \times y_v(q)) \qquad (1)$$

where $\ell_{\log}$ is the logistic loss function,

$$\ell_{\log}(z) = \log(1 + \exp(-z)) \qquad (2)$$

We adopt GBDT because it is able to model complex feature interactions and has been effective for other tasks such as text categorization [13] and rank-learning [22].

5. PORTABILITY

A *portable* vertical selection model is defined as one that can make vertical relevance predictions with respect to any arbitrary vertical. In other words, a portable model is not specific to a particular vertical, but rather agnostic of the candidate vertical being questioned for relevance.

Let us examine the distinction between a portable and non-portable vertical selection model with an example. Consider a single-evidence model that predicts a vertical relevant based on the number of times the query was issued previously to the vertical by users. This type of evidence is likely to be positively correlated with the relevance of the vertical in question. In fact, it is likely to be positively correlated with vertical relevance irrespective of the particular candidate vertical. On the other hand, consider a model that predicts a vertical relevant if the query is classified as related to the *travel* domain. This model may be effective at predicting the relevance of a vertical that serves travel-related content. However, it is probably not effective on a vertical that focuses on a different domain. This model is less portable.

Most existing single-evidence resource selection models can be considered portable [18, 17, 16, 15, 19]. For example, ReDDE [17] prioritizes resources for selection based on the estimated number of relevant documents in the collection. This expectation is a function of the number of documents sampled from the collection that are predicted relevant and the estimated size of the original collection. The greater the expectation the greater the relevance irrespective of the particular resource.

5.1 Basic Model

The objective of a portable vertical selection model, $f_\star$, is to maximize the average performance across source verticals. Our assumption is that if $f_\star$ performs consistently well across $\mathcal{S}$, then $f_\star$ will perform well on a new (target) vertical t. In general, the portability of a model is defined by a metric that quantifies performance for a vertical $s \in \mathcal{S}$ and a function that aggregates performance across verticals in $\mathcal{S}$.

For example, the portability, π, which uses the arithmetic mean of the logistic loss metric is defined by,

$$\pi^{\text{avg}}_{\log}(f_\star, y_\mathcal{S}, \mathcal{Q}_\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mu_{\log}(f_\star, y_s, \mathcal{Q}_s) \tag{3}$$

where $\mathcal{Q}_s$ is the set of training queries for source s and $\mathcal{Q}_\mathcal{S}$ is the set of those sets; similarly, y_s provides labels for vertical s and $y_\mathcal{S}$ is the set of these functions. We refer to the model which optimizes $\pi^{\text{avg}}_{\log}$ as the basic model. Notice that,

$$\pi^{\text{avg}}_{\log}(f_\star, y_\mathcal{S}, \mathcal{Q}_\mathcal{S}) = -\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} \ell_{\log}(f_\star(\phi_s(q)) \times y_s(q))$$

As a result, the solution which maximizes $\pi^{\text{avg}}_{\log}$ is equivalent to the solution which minimizes the logistic loss across all feature-vector/relevance-label pairs from all source verticals. That is, we can perform standard GBDT training on a pooling of each source vertical's training set.

5.2 Vertical Balancing

In the basic model's training set, positive instances correspond to relevant query-vertical pairs from *all* source verticals. For this reason, we expect the basic model to focus on evidence that is consistently predictive of relevance across source verticals, and hence predictive of the target vertical. In other words, vertical-specific evidence that is conflicting with respect to the positive class should be ignored. The challenge, however, is that the positive instances in the basic model's training pool may be skewed towards the more popular source verticals. This is problematic if these verticals are reliably predicted relevant using vertical-specific evidence, not likely to be predictive of the new target vertical. To compensate for this, we consider a *weighted* average of metrics across verticals. Specifically,

$$\pi^{\text{wavg}}_{\log}(f_\star, y_\mathcal{S}, \mathcal{Q}_\mathcal{S}) = \frac{1}{\mathcal{Z}} \sum_{s \in \mathcal{S}} w_s \mu_{\log}(f_\star, y_s, \mathcal{Q}_s) \tag{4}$$

where $\mathcal{Z} = \sum_{s \in \mathcal{S}} w_s$. We use the simple heuristic of weighting a vertical with the inverse of its prior,

$$w_s = \frac{1}{p_s}$$

where p_s is the prior probability of observing a query with relevant vertical s. This value is approximated with the training data,

$$p_s \approx \frac{\sum_{q \in \mathcal{Q}_s} y_s(q)}{|\mathcal{Q}_s|}$$

The goal is to make training instances from minority verticals more influential and those from majority verticals less.

It is easy to see that Equation 4 is a generalization of Equation 3. Because we use logistic loss, this technique reduces to training with an instance weighted logistic loss where the instances are weighted by w_s, the weight of the vertical,

$$\pi^{\text{wavg}}_{\log}(f_\star, y_\mathcal{S}, \mathcal{Q}_\mathcal{S}) = -\frac{1}{\mathcal{Z}} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} w_s \ell_{\log}(f_\star(\phi_s(q)) \times y_s(q))$$

As with the basic model, we can use standard GBDT training to optimize for this metric.

5.3 Feature Weighting

An alternative to optimizing for a portable model is to find portable features and to train a model using only those. A portable feature is defined as a feature which is highly correlated with relevance across all verticals. Recall that, across verticals, all features are identically indexed. Let ϕ^i be a predictor based only on the value of feature i. In previous work, the effectiveness of features across verticals was shown to be very dependent on the vertical being considered. In order to address the expected instability of feature predictiveness across verticals, we adopt a harmonic average for our aggregation method.

$$\pi^{\text{havg}}(\phi^i, y_\mathcal{S}, \mathcal{Q}_\mathcal{S}) = \frac{|\mathcal{S}|}{\sum_{s \in \mathcal{S}} \frac{1}{\mu(\phi^i_s, y_s, \mathcal{Q}_s)}} \tag{5}$$

Additionally, features, on their own, are not scaled to the label range, making the use of logistic loss difficult. Instead of constructing a mapping from a feature value to the appropriate range, we adopt a rank-based metric. Let $\rho_f(\mathcal{Q})$ be the ranking of $\mathcal{Q}$ by f. We use average precision as our rank-based metric,

$$\mu_{\text{AP}}(f, y, \mathcal{Q}) = \sum_{r=1}^{|\mathcal{Q}|} y(\rho_f(\mathcal{Q})_r) \times \mathcal{P}_r(y, \rho_f(\mathcal{Q})) \tag{6}$$

where $\rho_f(\mathcal{Q})_k$ denotes the query at rank k and $\mathcal{P}_k$ is the precision at rank k,

$$\mathcal{P}_k(y, \rho) = \frac{1}{k} \sum_{r=1}^{k} y(\rho_k)$$

In other words, for each feature, we rank queries by feature value and compute the harmonic mean average precision across verticals.[3] Having computed the portability of each feature, we build a portable model by restricting our training to the most portable features.

The most portable features were selected by inspecting the distribution of portability values. Because portability values are in the unit range, we model our data with the Beta distribution. We fit the Beta distribution using the method of moments and then select features whose portability is in the top quartile of this distribution.

6. ADAPTABILITY

Above, we focus on ways of improving the portability of a model by influencing the model to ignore evidence that is vertical-specific. The argument is that a model that focuses heavily on vertical-specific evidence will not generalize well to a new target vertical.

Given access to target-vertical training data, previous work reveals two meaningful trends [2]. First, given a wide-range of input features, most features contribute significantly to performance. In Arguello *et al.* [2], no small subset of features was solely responsible for effective vertical prediction. Second, the features that contributed the most to performance, which characterize the domain of the query, seem to be vertical-specific (assuming that verticals focus on different domains). Based on these observations, while ignoring vertical-specific evidence seems necessary to improve a

[3]Because we do not know whether the feature value has a positive or negative relationship with the label, we compute $\pi^{\text{havg}}_{\text{AP}}(f, y_\mathcal{S}, \mathcal{Q}_\mathcal{S})$ using ρ induced in both directions and use the max.

model's portability, a model customized to a particular vertical is likely to benefit from it.

In the context of adaptation for web search, Chen *et al.* [5] propose several ways to adapt an already-tuned GBDT model given data in a new domain. Their approach, Tree-based Domain Adaptation (TRADA), essentially consists of continuing the GBDT training process on labeled data from the target domain. More specifically, a set of new regression trees are appended to the existing model while minimizing a loss function (logistic loss, in our case) on the target data.

In our case, the challenge of using TRADA to adapt a model to a specific target is that we lack target-vertical training data. In the context of semi-supervised learning, self-training or bootstrapping [21] is the process of re-training a model using previous model predictions on unlabeled data. We combine self-training and model adaptation in the following way. First, we use a portable model to label a set of queries with respect to the target vertical. Then, we use TRADA to adapt the portable model to its own target-vertical predictions.

Tree adaptation provides a method for adjusting the modeling of all features. Just as we can select portable features for a portable model, we can select vertical-specific, non-portable features for adapting a model to a specific target vertical. That is, the base model may focus on portable features while the additional trees—added in the context of pseudo-training data—may focus on non-portable features. In this case, we use the same feature portability measure (Equation 5) but select the *least portable* features for tree augmentation.

Pseudo-labels for the target vertical were produced using the portable model's prediction confidence value with respect to the target vertical. We used the simple heuristic of considering the top $N\%$ most confident predictions as positive examples and the botton $(100 - N)\%$ predictions as negative examples.

7. METHODS AND MATERIALS

The objective is to predict the relevance of a target vertical given a query. For this reason, we evaluate on a per-vertical basis. Given a set of verticals $\mathcal{V}$ with query-vertical relevance labels, each vertical was artificially treated as the new target vertical and all remaining verticals as the source verticals. That is, for each $t \in \mathcal{V}$, we set $\mathcal{S} = \mathcal{V} - t$.

Given a set of queries with vertical-relevance judgements, evaluation numbers were averaged across 10 cross-validation test folds, using the remaining 90% of data for training. GBDTs require tuning several parameters: number of trees, maximum number of nodes per tree, and shrinkage factor (see Friedman [9] for details). These were tuned using a grid search and 10-fold cross-validation on each training fold. In all cases, cross-validation folds were split randomly. Significance is tested using a 2-tailed unpaired t-test.

TRADA was self-trained using predictions made on the test set. More specifically, at each cross-validation step, a basic model was tuned on the training fold (90% of all queries) and applied to the test fold (10% of all queries). Then, a TRADA model was pseudo-trained using predictions on the test fold. In other words, for a given vertical, we trained 10 basic and 10 TRADA models. Rather than tune pseudo-training parameter N, we present results for $N = 2.5, 5, 10\%$.

Table 1: Vertical descriptions.

vertical	retrievable items
finance	financial data and corporate information
games	hosted online games
health	health-related articles
images	online images
jobs	job listings
local	business listings
movies	movie show times
music	musician profiles
news	news articles
travel	travel and accommodation reviews and listings
video	online videos

7.1 Data

We focus on 11 verticals, described in Table 1. These verticals differ in terms of number of documents, domain (e.g., health, finance, travel), document type (e.g., news stories, embedded video clips, images), and level of query traffic.[4] Our evaluation data consists of 25,195 unique queries, randomly sampled from Web traffic. Given a query, human editors were instructed to assign verticals to one of four relevance grades ('most relevant', 'highly relevant', 'relevant', 'not relevant') based on their best guess of the user's vertical intent. It is possible for a query to have multiple verticals tied for a particular relevance grade. For about 25% of queries *all* verticals were labeled 'not relevant'. These are queries for which a user would prefer to see only Web search results.

7.2 Metrics

We are interested in the accuracy of a model's target-vertical predictions. Given a set of predictions for a target vertical, precision and recall can be computed by setting a threshold on the prediction confidence value. Rather than report a single precision-recall operating point, we evaluate using ranking metrics. These metrics are computed from a ranking of test queries by descending order of prediction confidence value, the probability that the target vertical is relevant to the query.

We adopt two rank-based metrics: average precision (AP) and normalized discounted cumulative gain (NDCG). In computing AP, the labels 'most relevant', 'highly relevant', and 'relevant' were collapsed into a single grade: relevant. We then compute average precision according to Equation 6.

NDCG differs from AP in two respects. First, it differentiates between relevance grades. Second, given a target vertical, t, it favors a model that is more confident on queries for which t is more relevant. Put differently, compared to AP, it punishes high-confidence errors more severely than low-confidence errors. Following Järvelin and Kekäläinen [12], NDCG for a target vertical t, evaluated over queries $\mathcal{Q}_t$, is computed as,

$$\mu_{\text{NDCG}}(f, y_t, \mathcal{Q}_t) = \frac{1}{\mathcal{Z}} \sum_{r=1}^{|\mathcal{Q}_t|} \frac{2^{y_t(\rho_f(\mathcal{Q}_t)_r)} - 1}{\log(\max(r, 2))} \qquad (7)$$

where y maps the relevance grade to a scalar ('most relevant': 3, 'highly relevant': 2, 'relevant': 1, 'not relevant': 0). The normalizer $\mathcal{Z}$ is the DCG of an optimal ranking of queries with respect to the relevance.

[4]Each vertical is associated with its own search interface.

Table 2: Target-trained ("cheating") results: AP and NDCG.

vertical	AP	NDCG
finance	0.556	0.861
games	0.741	0.919
health	0.800	0.945
hotjobs	0.532	0.814
images	0.513	0.855
local	0.684	0.926
movies	0.575	0.851
music	0.791	0.934
news	0.339	0.748
travel	0.797	0.947
video	0.290	0.701

Recall that our objective is to achieve the best performance possible without target-vertical training data. A major motivation is to alleviate the need for a model trained on target vertical data. Therefore, it is useful to measure our performance as a fraction of the performance achievable by a model with access to target training data. We show AP and NDCG results given *human-annotated* target-vertical training data in Table 2. A target-specific model (using all available features) was trained and tested for each vertical using 10-fold cross-validation. As in all results, we present performance averaged across test folds. Given our objective, this can be considered a "cheating" experiment. However, these numbers present a kind of upper bound for our methods. Our goal is to approximate these numbers using no target-vertical training data. For this reason, all results beyond this section are normalized by these numbers (i.e., results are reported as percentage of target-trained performance). Also, this normalization facilitates an understanding for the cost-benefit of labeling the new vertical given source-vertical labels and our proposed methods.

7.3 Unsupervised Baselines

Section 4.1.1 describes several query-vertical features that are used as input signals to our models. Prior work shows that each of these can be used as an unsupervised single-evidence vertical predictor [2, 1]. In other words, these methods can make target vertical predictions without training data. To justify the added complexity of our models, we compare against these single-evidence approaches. To conserve space, we present results for that which performed best in this evaluation: Soft.ReDDE [2]. Soft.ReDDE (the version for which the vertical language model was derived from vertical samples) performed equal to or better than the next best single-evidence method for all but 3/11 verticals based on both AP and NDCG.

8. RESULTS

We present portability results in Table 3. Across metrics, both vertical balancing (VB) and feature weighting (FW), that is, using only the most portable features, improves the performance of the basic model. Performance across verticals was either statistically indistinguishable or better. Compared to each other, feature weighting significantly improves the basic model across more verticals (8/11 for both metrics). Compared to Soft.ReDDE, the only method that does noticeably better is the basic model with feature weighting. Performance was significantly better for 4 verticals based on AP and 5 based on NDCG. Performance was significantly worse for only one vertical based on AP and no vertical based on NDCG.

Table 3: Portability results: normalized AP and NDCG. A $\blacktriangle$($\blacktriangledown$) denotes significantly better(worse) performance compared to Soft.ReDDE (SR). A $\vartriangle$($\triangledown$) denotes significantly better(worse) performance compared to the unbalanced basic model with all features. Significance was tested at the $p < 0.05$ level.

(a) AP

vertical	SR	basic (all feats.)	basic+VB (all feats.)	basic+FW (only portable feats.)
finance	0.446	$0.209^{\blacktriangledown}$	$0.199^{\blacktriangledown}$	$0.392^{\vartriangle}$
games	0.720	0.636	0.724	0.683
health	0.823	0.797	0.793	0.839
hotjobs	0.155	0.193	$0.226^{\blacktriangle}$	$0.321^{\blacktriangle\vartriangle}$
images	0.283	$0.365^{\blacktriangle}$	$0.404^{\blacktriangle}$	$0.390^{\blacktriangle}$
local	0.696	$0.543^{\blacktriangledown}$	$0.614^{\blacktriangledown\vartriangle}$	$0.628^{\blacktriangledown\vartriangle}$
movies	0.477	$0.294^{\blacktriangledown}$	$0.388^{\blacktriangledown\vartriangle}$	$0.478^{\vartriangle}$
music	0.757	$0.673^{\blacktriangledown}$	$0.700^{\blacktriangledown}$	$0.780^{\vartriangle}$
news	0.559	$0.293^{\blacktriangledown}$	$0.434^{\blacktriangledown\vartriangle}$	$0.548^{\vartriangle}$
travel	0.487	$0.571^{\blacktriangle}$	$0.618^{\blacktriangle\vartriangle}$	$0.639^{\blacktriangle\vartriangle}$
video	0.525	0.449	0.539	$0.691^{\blacktriangle\vartriangle}$

(b) NDCG

vertical	SR	basic (all feats.)	basic+VB (all feats.)	basic+FW (only portable feats.)
finance	0.776	$0.663^{\blacktriangledown}$	$0.651^{\blacktriangledown}$	$0.775^{\vartriangle}$
games	0.910	0.884	0.918	0.903
health	0.953	0.950	0.946	0.960
hotjobs	0.563	0.583	0.607	$0.671^{\blacktriangle\vartriangle}$
images	0.712	0.745	$0.776^{\blacktriangle}$	$0.768^{\blacktriangle}$
local	0.905	$0.875^{\blacktriangledown}$	$0.897^{\vartriangle}$	$0.910^{\vartriangle}$
movies	0.775	$0.685^{\blacktriangledown}$	0.745	$0.798^{\vartriangle}$
music	0.937	0.922	0.922	$0.957^{\blacktriangle\vartriangle}$
news	0.852	$0.703^{\blacktriangledown}$	$0.781^{\blacktriangledown\vartriangle}$	$0.875^{\vartriangle}$
travel	0.846	$0.881^{\blacktriangle}$	$0.908^{\blacktriangle\vartriangle}$	$0.911^{\blacktriangle\vartriangle}$
video	0.817	0.816	0.828	$0.902^{\blacktriangle\vartriangle}$

We present adaptability results in Table 4. TRADA adapts a basic model using its target-vertical predictions as pseudo-training data. Given its superior performance (Table 3), we used the unbalanced basic model with only portable features (basic+FW). We refer to this as the *base* model. TRADA was tested under two conditions. In the first condition, TRADA is given access to all features for adaptation. In the second condition, it is restricted access to *only* the non-portable features (those purposely ignored to improve the portability of the base model).

Table 4 presents several meaningful results. First, TRADA performs poorly when the adapted model is given access to all features (columns 4-6). In contrast, when restricted access to only the non-portable features (TRADA+FW), results improve (columns 7-9).

TRADA+FW performs either equal to or better than the base model in all but one case for AP and in *all* cases for NDCG. Similarly, across metrics, TRADA+FW significantly outperforms Soft.ReDDE across most verticals for all values of N. It performs significantly worse than Soft.ReDDE in three cases in terms of AP and none in terms of NDCG. Overall, we interpret this as a positive result in favor of adaptability. TRADA succeeds at adapting a portable model to a specific target vertical at no additional editorial cost.

Table 4: Trada results: normalized AP and NDCG. A $\blacktriangle$($\blacktriangledown$) denotes significantly better (worse) performance compared to soft.redde (SR). A $\vartriangle$($\triangledown$) denotes a significantly better(worse) performance compared to the base model. Significance was tested at the $p < 0.05$ level.

(a) AP

			trada			trada+FW		
		basic+FW	(all features)			(only non-portable feats.)		
vertical	SR	(only portable feats.)	(N=2.5%)	(N=5%)	(N=10%)	(N=2.5%)	(N=5%)	(N=10%)
finance	0.446	0.392	0.364	$0.328^{\blacktriangledown}$	$0.226^{\blacktriangledown\triangledown}$	0.476	0.407	$0.339^{\blacktriangledown}$
games	0.720	0.683	0.735	0.660	$0.491^{\blacktriangledown\triangledown}$	$0.819^{\blacktriangle\vartriangle}$	$0.817^{\blacktriangle\vartriangle}$	$0.787^{\vartriangle}$
health	0.823	0.839	0.814	0.813	$0.592^{\blacktriangledown\triangledown}$	$0.907^{\blacktriangle\vartriangle}$	$0.868^{\blacktriangle}$	0.818
hotjobs	0.155	$0.321^{\blacktriangle}$	$0.360^{\blacktriangle}$	$0.384^{\blacktriangle}$	$0.345^{\blacktriangle}$	$0.390^{\blacktriangle}$	$0.348^{\blacktriangle}$	$0.323^{\blacktriangle}$
images	0.283	$0.390^{\blacktriangle}$	$0.320^{\triangledown}$	$0.370^{\blacktriangle}$	$0.405^{\blacktriangle}$	$0.410^{\blacktriangle}$	$0.499^{\blacktriangle\vartriangle}$	$0.523^{\blacktriangle\vartriangle}$
local	0.696	$0.628^{\blacktriangledown}$	$0.523^{\blacktriangledown\triangledown}$	$0.601^{\blacktriangledown}$	$0.609^{\blacktriangledown}$	$0.562^{\blacktriangledown\triangledown}$	$0.614^{\blacktriangledown}$	0.663
movies	0.477	0.478	0.493	0.462	$0.411^{\triangledown}$	$0.640^{\blacktriangle\vartriangle}$	$0.587^{\blacktriangle\vartriangle}$	$0.578^{\blacktriangle\vartriangle}$
music	0.757	0.780	0.751	0.778	0.760	$0.868^{\blacktriangle\vartriangle}$	$0.866^{\blacktriangle\vartriangle}$	$0.838^{\blacktriangle\vartriangle}$
news	0.559	0.548	0.509	0.556	0.523	0.607	$0.665^{\blacktriangle\vartriangle}$	0.615
travel	0.487	$0.639^{\blacktriangle}$	$0.531^{\triangledown}$	$0.573^{\blacktriangle\triangledown}$	$0.597^{\blacktriangle}$	$0.744^{\blacktriangle\vartriangle}$	$0.709^{\blacktriangle\vartriangle}$	$0.710^{\blacktriangle\vartriangle}$
video	0.525	$0.691^{\blacktriangle}$	0.633	0.648	0.586	$0.735^{\blacktriangle}$	$0.722^{\blacktriangle}$	$0.688^{\blacktriangle}$

(b) NDCG

			trada			trada+FW		
		basic+FW	(all features)			(only non-portable feats.)		
vertical	SR	(only portable feats.)	(N=2.5%)	(N=5%)	(N=10%)	(N=2.5%)	(N=5%)	(N=10%)
finance	0.776	0.775	0.760	0.718	$0.632^{\blacktriangledown\triangledown}$	0.826	0.765	0.734
games	0.910	0.903	0.920	0.885	$0.778^{\blacktriangledown\triangledown}$	$0.947^{\blacktriangle\vartriangle}$	$0.951^{\blacktriangle\vartriangle}$	$0.938^{\vartriangle}$
health	0.953	0.960	0.947	0.939	$0.827^{\blacktriangledown\triangledown}$	$0.974^{\blacktriangle}$	0.960	0.953
hotjobs	0.563	$0.671^{\blacktriangle}$	$0.720^{\blacktriangle}$	$0.736^{\blacktriangle}$	$0.710^{\blacktriangle}$	$0.747^{\blacktriangle}$	$0.698^{\blacktriangle}$	$0.676^{\blacktriangle}$
images	0.712	$0.768^{\blacktriangle}$	0.745	$0.775^{\blacktriangle}$	$0.790^{\blacktriangle}$	$0.801^{\blacktriangle}$	$0.850^{\blacktriangle\vartriangle}$	$0.869^{\blacktriangle\vartriangle}$
local	0.905	0.910	$0.885^{\blacktriangledown\triangledown}$	0.907	0.898	0.901	0.911	$0.930^{\blacktriangle\vartriangle}$
movies	0.775	0.798	0.808	0.790	$0.732^{\triangledown}$	$0.856^{\blacktriangle}$	$0.842^{\blacktriangle}$	$0.840^{\blacktriangle}$
music	0.937	$0.957^{\blacktriangle}$	0.939	0.939	0.931	$0.977^{\blacktriangle\vartriangle}$	$0.974^{\blacktriangle\vartriangle}$	$0.965^{\blacktriangle}$
news	0.852	0.875	0.850	0.868	0.828	0.898	0.908	0.879
travel	0.846	$0.911^{\blacktriangle}$	$0.875^{\triangledown}$	$0.890^{\blacktriangle}$	$0.889^{\blacktriangle}$	$0.944^{\blacktriangle\vartriangle}$	$0.933^{\blacktriangle\vartriangle}$	$0.926^{\blacktriangle}$
video	0.817	$0.902^{\blacktriangle}$	0.869	0.865	0.827	$0.930^{\blacktriangle}$	$0.896^{\blacktriangle}$	0.880

9. DISCUSSION

In the previous section, we demonstrated that the performance improvement for both portable and adaptive models requires measuring individual feature portability. In order to investigate precisely which features were being selected, we plotted the value of $\pi_{\mathrm{AP}}^{\mathrm{havg}}$ in Figure 1. As it turns out, the same five features were consistently chosen as the most portable for all target verticals (i.e., for all sets of source verticals). Interestingly, these correspond to our five query-vertical features (Section 4.1.1). Conversely, those features which were the least portable were consistently our remaining query features (Section 4.1.2). In hindsight, this observation makes sense. Many existing resource selection methods score resources using a single metric and focus exclusively on query-vertical evidence [4, 15, 18, 17, 16, 19]. Despite this observation, we recommend future experiments continue to measure feature portability since this behavior may not generalize to different sets of verticals and different tasks.

Vertical balancing significantly improved the basic model only across 4/11 verticals based on AP and 3/11 based on NDCG. Recall that we introduced balancing in order to discourage examples from source verticals with high priors dominating the training set. However, this does not address cases where several sources have the same non-portable features correlated with relevance. For example, *video* and *images* tend to be relevant to queries that mention a celebrity name; *travel* and *local* tend to be relevant to queries that mention a geographic entity; and *finance* and *jobs* tend to be relevant to queries that contain a company name.

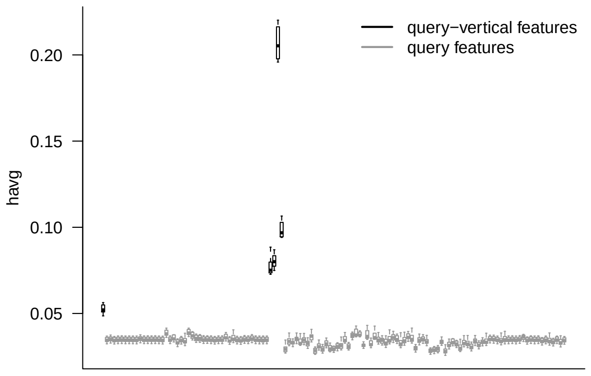

Figure 1: Feature portability values (π^{hmap}) across sets of source verticals.

Even though vertical balancing addresses a single vertical's dominance in the training set, it does not address a small coalition of related verticals causing the model to use non-portable features. We believe that a more robust averaging technique—for example, the harmonic average used for feature portability—may result in superior performance in the presence of similar source verticals.

In the previous section, TRADA improves considerably when restricted access to *only* the non-portable features for adaptation. We believe this is due to the following. TRADA was pseudo-trained using predictions from the base model. This model (our best portable model) used only the most portable features. The set of all features is a superset of these. When TRADA is given access to the same features used by the base model, the adapted model tends to focus on these features in order to better fit the original base-model predictions. Therefore, the non-portable features (purposely inaccessible to the base model) were ignored. As it turns out, our set of non-portable features are highly effective given target-vertical training data. We compared a set of target-trained models using only the portable features (excluding the non-portable ones) with our target-trained models using all features (Table 2). When given access to non-portable features performance improved significantly across *all* verticals for AP and all but one vertical for NDCG (results suppressed due to space limitations). When restricted access to only the non-portable features for adaption, TRADA is forced to focus on these highly effective features. This mechanism can be seen as a sort of regularization ensuring that both portable and non-portable features are used for prediction.

With respect to pseudo-training data parameter N, we observe that the optimal value of N seems to be vertical-dependent. There may be two reasons for this. First, the base model performance (column 5 in Table 3) is also vertical-dependent. Given a fixed value of N across verticals, pseudo-labels from some verticals may be noisier than others. Second, the optimal value of N for a given vertical may correlate with the vertical's prior.

10. CONCLUSION

Maximizing the return on editorial investment is an important aspect of any system requiring training data. We presented an ensemble of approaches which significantly improve prediction of a new target vertical using only source-vertical training data. Our results demonstrate that *model portability*, the ability of a model to generalize across different target verticals, requires careful attention to *feature portability*, the ability of a *feature* to correlate with vertical relevance across different target verticals. We found that those features which seemed to be the most portable—and hence most important for a portable model—were query-vertical features as opposed those that are independent of the candidate vertical. Conversely, when we tried to adapt a model for a specific target, the least portable features appeared to be those most important for the adapted model to consider.

Furthermore, we showed that a portable solution can be used to build a target-specific one at no additional editorial cost. Our best approach adapted a portable model using its own target-vertical predictions. This approach consistently outperformed both the base model and a competitive alternative which does not require adaptation. Results also showed that, given available resources, human-annotation on the new target vertical remains the best alternative.

This work could be extended in several directions. In terms of portability, vertical balancing may be improved by modeling the similarity (in terms of predictive evidence) between source verticals. In terms of adaptability, further improvements may be achieved by modeling the similarity between each source vertical and the target vertical.

11. ACKNOWLEDGMENTS

We would like to thank Jing Bai, Daniel Boies, Hugues Bouchard, Jean-François Crespo, and Alexandre Rochette for helpful discussions and feedback. This work was supported in part by the NSF grants IIS-0916553 and IIS-0841275 and a generous gift from Yahoo! through its Key Scientific Challenges program. Any opinions, findings, conclusions, and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

12. REFERENCES

[1] J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In *CIKM 2009*, pages 1277–1286. ACM, 2009.
[2] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR 2009*, pages 315–322. ACM, 2009.
[3] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP 2006*, pages 120–128. ACL, 2006.
[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR 1995*, pages 21–28. ACM, 1995.
[5] K. Chen, R. Lu, C. K. Wong, G. Sun, L. Heck, and B. Tseng. Trada: tree based ranking function adaptation. In *CIKM 2008*, pages 1143–1152. ACM, 2008.
[6] H. Daumé III. Frustratingly easy domain adaptation. In *ACL 2007*, pages 256–263. ACL, 2007.
[7] F. Diaz. Integration of news content into web results. In *WSDM 2009*, pages 182–191. ACM, 2009.
[8] F. Diaz and J. Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *SIGIR 2009*, pages 323–330. ACM, 2009.
[9] J. H. Friedman. Gradient function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999.
[10] J. Jiang. *Domain Adaptation in Natural Language Processing*. PhD thesis, University of Illinois at Urbana-Champaign, 2008.
[11] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *ACL 2007*, pages 264–271. ACL, 2007.
[12] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *TOIS*, 20:2002, 2002.
[13] S.-M. Kim, P. Pantel, L. Duan, and S. Gaffney. Improving web page classification by label-propagation over click graphs. In *CIKM 2009*, pages 1077–1086. ACM, 2009.
[14] S. Satpal and S. Sarawagi. Domain adaptation of conditional probability models via feature subsetting. In *PKDD 2007*, pages 224–235. Springer-Verlag, 2007.
[15] J. Seo and B. W. Croft. Blog site search using resource selection. In *CIKM 2008*, pages 1053–1062. ACM, 2008.
[16] M. Shokouhi. Central rank based collection selection in uncooperative distributed information retrieval. In *ECIR 2007*, pages 160–172, 2007.
[17] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR 2003*, pages 298–305. ACM, 2003.
[18] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *CIKM 2002*, pages 391–397. ACM, 2002.
[19] P. Thomas and M. Shokouhi. Sushi: Scoring scaled samples for server selection. In *SIGIR 2009*. ACM, 2009.
[20] J. Xu and X. Li. Learning to rank collections. In *SIGIR 2007*, pages 765–766. ACM, 2007.
[21] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL 1995*, pages 189–196. ACL, 1995.
[22] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR 2007*, pages 287–294. ACM, 2007.