Identifying Facet Mismatches In Search Via Micrographs

Sriram Srinivasan
University of California Santa Cruz
ssriniv9@ucsc.edu

Nikhil S Rao
Amazon Inc.
nikhilsr@amazon.com

Karthik Subbian
Amazon Inc
ksubbian@amazon.com

Lise Getoor
University of California Santa Cruz
getoor@ucsc.edu

ABSTRACT

E-commerce search engines are the primary means by which customers shop for products online. Each customer query contains multiple *facets* such as product type, color, brand, etc. A successful search engine retrieves products that are relevant to the query along each of these attributes. However, due to lexical (erroneous title, description, etc.) and behavioral irregularities (clicks or purchases of products that do not belong to the same facet as the query), some mismatched products are shown in the search results. These irregularities are often detected using simple binary classifiers like gradient boosted decision trees or logistic regression. Typically, these binary classifiers use strong independence assumptions between the samples and ignore structural relationships available in the data, such as the connections between products and queries. In this paper, we use the connections that exist between products and query to identify a special kind of structure we refer to as a *micrograph*. Further, we make use of Statistical Relational Learning (SRL) to incorporate these micrographs in the data and pose the problem as a structured prediction problem. We refer to this approach as *structured mismatch classification* (SMC). In addition, we show that naive addition of structure does not improve the performance of the model and hence introduce a variation of SMC, strong SMC (S^2MC), which improves over the baseline by passing information from high-confidence predictions to lower confidence predictions. In our empirical evaluation we show that our proposed approach outperforms the baseline classification methods by up to 12% in precision. Furthermore, we use quasi-Newton methods to make our method viable for real-time inference in a search engine and show that our approach is up to 150 times faster than existing ADMM-based solvers.

1 INTRODUCTION

When customers shop online, they issue queries that describe their intent along multiple facets of their desired product: brand, color, product-type, age group, size, gender, and activity. For example, the query "red adidas shorts for boy age 6" has age, gender, color, brand and a product type specified. These facets are critical for matching, ranking, and navigation. For instance, in the case of navigation, a customer might first look for a specific brand and then narrow down their choices based on color, obtaining relevant results that align closely with their intent. However, due to past behavioral associations (such as clicks, purchases, and cart-adds) or noisy lexical information (such as low-quality seller supplied keywords), or competition between brands, the search might result in mismatched products.

Consequently, identifying facet mismatches between a query and products in the catalog to avoid displaying irrelevant results is an important component of providing customers with a satisfying shopping experience. A typical model for recognizing facet mismatches outputs a score for one or more facets given a query-product pair. This score indicates whether the product is a good match for the query along that specific facet.

In modern datasets, there are vast amounts of additional structural information about queries, products, and their relationships. This additional information can manifest itself in several ways and can be used as side information during the retrieval, ranking and mismatch classification training process.

(1) Products are typically co-purchased or co-viewed together. We can include this information as a product-product graph.
(2) We can generate query and product latent representations (embeddings) and use cosine similarity as an affinity score between (query, product), (query, query), and (product, product) pairs.
(3) Customer query reformulations within the same session can be used to compute query-query similarities.

Incorporating such structural side information typically yields a boost in model performance. Indeed, learning with side information has shown to be successful in several applications such as recommender systems [22, 30], knowledge graphs [25], entity resolution [29], computer vision [15], and has recently been applied even in deep learning tasks [34, 36].

Many information retrieval [2, 7, 21, 26] and ranking [35, 37] tasks have used the structural information to improve their models. These approaches create a graph (or similar relational structure) using an item's lexical information and improve the list of items retrieved for a specific query. However, in this work, we do not focus on using structural information to retrieve a better list of items, rather we show a way to improve the quality of the retrieved list by identifying mismatched items. Our approach focuses on using heterogeneous structural information to identify search mismatches which can be subsequently used to either reorder or improve the list by replacing the mismatched items.

CIKM'19, November 2019, Bejing, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Statistical relational learning (SRL) frameworks [9, 13] are an effective way of incorporating heterogeneous structural information to make more accurate predictions. An SRL model is typically defined through the use of a set of weighted logical rules that define a probabilistic graphical model. A recent SRL framework called *probabilistic soft logic* (PSL) generates a specific type of probabilistic graphical model known as a *Hinge-Loss Markov Random Field* (HL-MRF) [3] that has shown state-of-the-art performance in several domains by augmenting heterogeneous structural information [22, 32]. A key property of PSL is that the random variables generated for the corresponding HL-MRF are continuous between zero and one. This casts the inference task into a convex optimization problem, on which a scalable solver, the *Alternating Directions Method of Multipliers* (ADMM) [6], is used to obtain state of the art performance.

However, using additional information and performing graphical model inference is slower than performing inference using a simple pointwise binary classifier for facet mismatch. This presents an additional challenge for applying these ideas to product search, which requires the use of models with extremely fast inference for real-time retrieval of search results. Existing scalable approaches for graphical model inference [10, 33] do not meet the latency constraints of search systems. A customer may abandon the search if it takes more than a few milliseconds between typing in a query and obtaining the results. This constraint makes it extremely difficult to use additional side information or use sophisticated models with higher computational complexity.

In this paper, we develop a novel approach using the PSL framework for facet mismatch classification and apply it to the task of detecting product-type mismatches, a particularly egregious form of facet mismatch since they lead to a significantly degraded customer experience. As an example, a customer searching for an "iPhone" expects to see different variety of iPhones in the search results and not iPhone case or a screen protector. We show that incorporating additional structural information present in the data can significantly improve the classification performance. Secondly, to tackle the problem of near real-time inference, we cast the problem of PSL optimization as minimizing an SVM-like objective function and use a *Trust Region Quasi-Newton* (TRON) [24] method to solve it. We show that the resulting method achieves orders of magnitude speedups over existing approaches for PSL optimization. Note that the approach we propose is quite general and can be applied to many different label propagation application, but in this paper we focus on detecting product-type mismatches in search.

1.1 Contributions and Organization

To summarize, the contributions of our paper are as follows:

- We introduce a special query-product relationship graph that we refer to as a *micrograph* which we show can be used to improve facet mismatch classifiers. Micrographs ensure that our approach scales independently of the number of queries, allowing us to use it for industry-sized datasets.
- We show how micrographs can be utilized in the PSL framework to improve facet mismatch classification by performing collective inference. We refer to this approach as *structured mismatch classification* (SMC). We also show that naive inclusion of structure does not improve the model performance significantly. Further, we introduce a variant of SMC which we refer to as *strong SMC* (S^{2}MC) which selectively performs joint inference to improve overall mismatch identification.
- We perform extensive experiments across multiple datasets and show that the method we propose improves upon baseline methods in performance by up to *12%* increase in precision and *11%* increase in F1 scores.
- We reformulate the resulting optimization problem which enables us to perform near real-time inference using quasi-Newton methods. Through a series of experiments, we show that our approach is scalable and can be used to make real-time predictions. Our approach of using a quasi-Newton method yields up to *150X speedup* over the existing solver (ADMM).

The rest of the paper is organized as follows. In Section 2, we formally set up the problem and discuss traditional solutions to the problem. Next in Section 3, we introduce the concept of micrographs, elaborate on them and show how they can be used in our problem setting. In Section 4 we give some background on HL-MRFs and PSL and in Section 5 we define our approach on using micrographs to perform collective inference. Next, in Section 6, we discuss the need for extremely fast inference and show how we can efficiently make predictions using trust region Newton methods, which yields orders-of-magnitude speedups over existing ADMM solvers. We perform extensive experiments on multiple datasets and their results in Section 7. Finally, we summarize and conclude the paper in Section 8.

2 PROBLEM DEFINITION AND TRADITIONAL APPROACH

Our task is to improve search results by identifying the products whose facets do not match that of a query. In this section, we formally define this task as *facet mismatch classification* and discuss some traditional approaches to address this problem.

2.1 Facet mismatch classification

The *facet mismatch classification* is the general task of classifying a (query, product) pair as matched (or relevant) along one or more facets. Formally, we define facet mismatch classification for a single facet as follows. Note that generalizing this definition for more than one facet is straightforward.

Definition 2.1 (Facet mismatch classification). Consider a set of all possible queries $\mathcal{Q}$ and a set of all possible products $\mathcal{P}$. Given a query $q \in \mathcal{Q}$ and a relevance model M such that $M(q) = \mathbf{p}_q$ where $\mathbf{p}_q$ is a ranked list of products returned as relevant to the query q by the model M. Let f be a facet so that $f(q)$ and $f(p_q^i)$ are indicator variables for the facet being present in the query, and the i^{th} product in $\mathbf{p}_q$. Then p_q^i is a facet mismatch if $\gamma_{q,p_q^i} := \mathbb{1}(f(p_q^i) \neq f(q)) = 1$, where $\mathbb{1}$ is an indicator function.

2.2 Traditional approach

The above mentioned problem can be seen as a classification problem with a task of predicting γ_{q,p_q^i} for any given (q, p_q^i) pair. As

facet mismatch is a more subtle classification problem than traditional relevance, one cannot simply use user logs and CTR to obtain a dataset. The data available for training will contain pairs of (q, p_q^i) along with a label γ_{q,p_q^i} where the labels are typically annotated by human curators. Human curation implies that the training datasets are typically much smaller than standard ranking datasets.

Any binary (or multilabel) classifier such as logistic regression, or deep neural networks [5] can be used to perform this task, with the caveat of lending itself to fast online predictions. We refer to using such traditional classifiers for performing facet mismatch classification as *traditional mismatch classification* (TMC). To be able to handle low latency, one can perform this classification task using an off-the-shelf industry workhorse model like Gradient Boosted Decision Tree (GBDT) [8, 12]. In particular, for search and information retrieval systems, GBDTs have been shown to handle both categorical and ordinal features with efficient training and fast real-time inference [17]. The GBDT models trained in search applications use a mix of text and behavioral features, depending on either the query, product or both. A set of joint features $(\varphi(q, p_q^i))$ is used to make prediction on the facet mismatch classification problem:

$$\rho_{q,p_q^i} = GBDT(\varphi(q, p_q^i)) \tag{1}$$

In this work we use GBDT models as our TMC. In later sections we show how we make use of this score (ρ_{q,p_q^i}) in our model. TMC uses this score to determine whether there is a facet mismatch, i.e., $\gamma_{q,p_q^i} = \mathbb{1}(\rho_{q,p_q^i} > t)$, where $t \in (0, 1)$ is a threshold.

3 RELATIONAL STRUCTURE AND MICROGRAPHS

The facet mismatch classification problem can also be seen as an edge labeling problem on a graph. Consider z queries $(q_1 \dots q_z)$ and j products $(p_1 \dots p_j)$ as nodes to the left and right side of a bipartite graph (see Figure 1a, here z and j are set to four). The existence of an edge between any query q and product p in this graph represents either textual or behavioral match between a (query, product) pair i.e., $p \in \mathbf{p}_q$. A solid edge between a (query, product) pair indicates an observed mismatch (human annotated to zero or one) and dotted edge indicates that the mismatch value need to be inferred. Our goal here is to infer whether an edge is a mismatch or not given a few edge labels. Consider an example query "black apple iphone", all "iphone" products match on the product-type facet and form an edge to the query with value zero. Other product types such as iphone cases and screen protectors have an edge to the query with value one, as they do not match on the product type facet. The label for some edges are known (manually labeled by human judges) and our task is to use the existing edge labels to infer the labels for the unknown edges.

While TMCs can be used to perform facet mismatch classification, they suffer from a major drawback. They assume strong conditional independence in the data (i.e., the value assigned for ρ_{q,p_q^i} is conditionally independent of other pairs in the data given q and p_q^i), and predict the edge labels. This assumption makes inference very scalable. However, it completely ignores the relationship between the query and the list of products for which we need to determine facet mismatch. We depict these additional relationships in Figure 1b. The edge between queries represents many possible relations, such as semantic similarities between queries, query intent relation, and so on. Similarly, the connection between products can be a lexical or semantic similarity and co-purchase behavior. Further, the predictions produced by TMC can also be represented as an edge between (query,product) and can be used as preliminary mismatch scores. The presence of additional edges makes the prediction task γ_{q,p_q^i} dependent on related products and queries. Therefore, a prediction ρ_{q,p_q^i} is no longer conditionally independent and joint predictions have to be performed. For instance, if for some query q, $\gamma_{q,p_q^1} = 1$, and p_q^1 and p_q^2 are connected to each other via a similarity edge, then we'd expect $\gamma_{q,p_q^2} = 1$ This implies that the label assigned for both the edges depend on each other and need to be predicted jointly. This form of classification is commonly known as *collective classification* [31].

There have been many approaches proposed in varied applications to perform collective classification [1, 20, 28, 31, 36]. The primary issue with such methods is a large amount of time required to perform inference as the complexity of the algorithm grows exponentially in the number of nodes and relations. The heterogeneity of the structural relationships adds to this complexity.

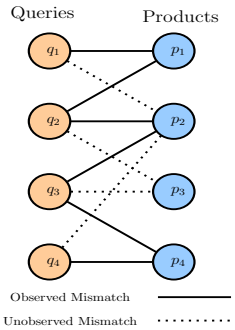
The models that perform collective classification are typically transductive in nature, meaning we need to perform a full (graphical model) inference for a given customer query at run-time. Furthermore, the queries themselves can be arbitrary and precomputing the results and serving them at runtime is not possible. These reasons have precluded the use of such methods for search and information retrieval tasks. We show that by carefully constructing graphs, we can perform inference at runtime. The idea is to break up the graph so that we can perform inference over several smaller (query independent) graphs in parallel. For this reason, we break up the graphs per query and consider (query, product) and (product, product) relationships. We refer to this smaller, more tractable graph as a *micrograph*. Figure 1c shows an illustration. Formally, we define a micrograph as follows:

Definition 3.1 (micrograph). A *micrograph* is a graph G, with the vertices being a query q and the list of top-k products $\mathbf{p}_q$ obtained through a retrieval model $M(q)$. The edges in the graph correspond to known (q, p_q^i) labels, and any product-product (p_q^i, p_q^j) edges.

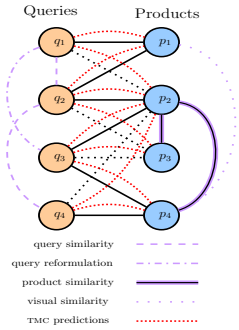
Typically, a customer does not scroll past a small number of items in response to a query. Hence, we focus our attention to a small k in the above definition, $\sim O(10)$. This allows us to use micrographs with very small number of nodes, making it possible to perform real-time inference.

In this work, we improve the predictions made by the TMC model using these micrographs. We can generate query-product edges as predicted by the TMC for any given (query,product) pair. However, some of these edges may be of low-confidence, or in some cases incorrect. There will also be edges between products, based on co-purchases or semantic similarities which can be computed for all product pairs. The task now is to perform a joint prediction on facet mismatch edges between query and product using all the above mentioned observed data in the micrograph.

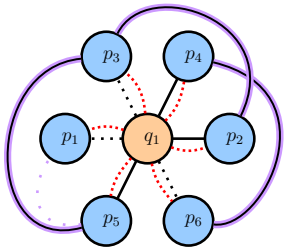
Given these micrographs that encode different information in the graph, the next step is to reason over this graph to improve



(a) Facet mismatch classification problem as edge labeling problem.



(b) Structured facet mismatch classification with multiple relational structure.



(c) A micrograph representation for a query.

Figure 1: The facet mismatch classification problem as a structure prediction problem. Black dotted edges represent unobserved facet matches. A black solid edge represents an observed facet mismatch and has a value of zero or one. The prediction task is to infer the values for the black dotted edges based on the available structural relationships (all other edges).

the facet mismatch score. We view this task as performing inference in a graphical model provided by the micrograph. A graphical model created using the micrographs will contain observed random variables as the observed edges in the micrograph and the unobserved random variables to infer are the facet mismatch edge between query and products. In Section 5, we elaborate SMC and s^2MC which combine the micrograph information and perform joint predictions through inference in a graphical model. We use PSL framework to generate the graphical model and perform inference. The PSL framework produces a specific type of graphical model called the *Hinge Loss Markov Random Field* (HL-MRF) [3]. An advantage of HL-MRFs is that we can cast the HL-MRF inference as a convex optimization program and use existing solvers to obtain an exact solution. In the following section, we discuss a brief review of HL-MRF and their relation to PSL.

4 HL-MRFS AND PSL

Probabilistic Soft Logic (PSL) [18] is a probabilistic programming language which can be used to generate large and complex graphical models. A model in PSL is defined through a set of weighted first-order logical rules (template rules). A small set of template rules when instantiated with data can generate large and complex graphical models. A key capability of a rule based PSL model is that they can efficiently combine multiple similarities and other connections in data to perform collective reasoning. Each predicate in a PSL rule, corresponds to a specific relation in the real world. For instance, product similarities can be represented using a predicate $Similarity(P_1, P_2)$ where P_1 and P_2 are placeholders for two products and a facet mismatch can be represented through a predicate $Mismatch(Q, P_1)$ where Q is a placeholder for a query. These predicates when instantiated with Q ="*iPhone*", P_1 ="*white iPhone case*", and P_2 ="*black iPhone case*" generate ground predicates $Similarity$("*white iPhone case*", "*black iPhone case*") and $Mismatch$("*iPhone*", "*white iPhone case*"). Each of these ground predicates represents a random variable in a graphical model, some are observed ($Similarity$) and some are unobserved ($Mismatch$). Note that the $Mismatch$ predicate could be partially observed as well. Similarly, every template rule when grounded with data produce cliques in a graphical model. A model is thus fully instantiated with data to produce all cliques of the graphical model. For example, here is a simple rule to connect mismatch scores through similarity edges:

$$w : Mismatch(Q, P_1) \wedge Similar(P_1, P_2) \rightarrow Mismatch(Q, P_2)$$

where w represents the importance or weight of the rule. When the above rule is instantiated it creates a clique (w : $Mismatch$("*iPhone*", "*white iPhone case*") $\wedge$ $Similar$("*white iPhone case*", "*black iPhone case*") $\rightarrow$ $Mismatch$("*iPhone*", "*black iPhone case*")) with three random variables (two unobserved and one observed). Collective inference can be performed on this ground graphical model to make joint predictions on the unobserved random variables. This makes the processes of incorporating complex structure into a model easy and interpretable.

A distinguishing trait of PSL is that the Boolean predicates are relaxed to be continuous in range $[0, 1]$. This continuous relaxation can be interpreted as a relaxation of logical MRFs [3]. Further, the truth values can be also be seen as rounding probabilities of Boolean predicates or a form of soft/fuzzy logic [3]. By viewing the ground rules as soft logic statements, in specific Lukasiewicz Logic [19], a potential associated with a clique generated by a ground logical rule can be expressed as hinge-loss functions. For example, when using Lukasiewicz Logic, $a \rightarrow b$ corresponds to the hinge function $max(a-b, 0)$ and $a \wedge b$ corresponds to $max(a+b-1, 0)$. These hinge-loss potentials form a special kind of graphical model called the HL-MRF. As HL-MRFs are composed of continuous random variables and hinge-loss potentials, they admit to efficient inference. Formally, an HL-MRF defines a conditional probability density function over unobserved random variables $\mathbf{Y}$ conditioned on observed random variables $\mathbf{X}$,

$$P(\mathbf{Y}|\mathbf{X}) \propto exp\Big(-\sum_{i=1}^{n} w_i \phi_i(\mathbf{Y}, \mathbf{X})\Big) \tag{2}$$

$$\phi_i(\mathbf{Y}, \mathbf{X}) = (max\{0, l_i(\mathbf{Y}, \mathbf{X})\})^{d_i}, d_i \in \{1, 2\} \tag{3}$$

where ϕ_i is a hinge-loss potential function, l_i is a linear function over $\mathbf{Y}$, and $\mathbf{X}$. The value of d_i determines if we want to use a hinge or a squared hinge loss and in this paper we use the latter as it gives us added smoothness. The variables in $\mathbf{X}$ and $\mathbf{Y}$ are in the unit interval $[0, 1]$. Each $w_i \in \mathbb{R}^+$ represents the importance of that specific potential which can be learned [4] or set manually, and n represents the total number of potentials. Note that Equation 3 is log-concave in $\mathbf{Y}$, so maximum a posteriori (MAP) inference to find the optimal $\mathbf{Y}$ in HL-MRFs can be solved exactly via convex optimization. The objective function is the following:

$$\underset{\mathbf{Y}}{\operatorname{argmax}} P(\mathbf{Y}|\mathbf{X}) = \underset{\mathbf{Y}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \phi_i(\mathbf{Y}, \mathbf{X}) \tag{4}$$

The above expression is usually solved using alternating direction method of multipliers (ADMM) [6] approach as described in [3].

5 STRUCTURED MISMATCH CLASSIFICATION

We can represent all relationships in a micrograph using PSL and create an HL-MRF to perform efficient inference at run-time. In this section, we define the specific relations that were used to improve facet mismatch classification. We begin by using the prediction scores produced by a TMC for a particular (query, product) edge and propagate this information to other related products. We construct (product, product) edges using the semantic similarity between them. We use latent representations of the products (based on their meta-data) to compute the similarity score. Further, we distinguish high-confidence scores to further improve the overall predictions in the micrograph.

5.1 Using TMC predictions

In the transductive setting, where making a prediction requires an inference step, we need a large set of edges with labeled data to use in a graphical model [10]. However, as explained in the previous sections, obtaining such large amounts of ground truth data is expensive and time-consuming. Instead, one can use the output of an existing discriminative model as the "seed" labels on the edges of the graph. In particular we make use of the scores produced by an underlying TMC, trained on existing ground truth information as labels. The following rules encode the TMC scores:

$$\text{TMC}(Q, P) \rightarrow \mathit{Mismatch}(Q, P) \tag{5}$$

$$\neg\text{TMC}(Q, P) \rightarrow \neg\mathit{Mismatch}(Q, P) \tag{6}$$

$\mathit{Mismatch}(Q, P) \in [0, 1]$ is the target predicate to be inferred. $\text{TMC}(Q, P) \in [0, 1]$ is the prediction score produced by the TMC for facet mismatch. The above rules incorporate the pairwise classifier which encodes the signal from multiple behavioural and lexical features of the query and product. Note that we can combine scores from multiple classifiers in this way, creating an ensemble of multiple TMCs. We restrict ourselves to a single underlying classifier here for ease of exposition. In our subsequent rules we make use of additional information to improve the predictions.

5.2 Using product similarities

We can propagate the product scores to similar products to perform joint inference. The primary idea is as follows: if a particular (query, product) pair is a facet mismatch, then substitutable products should also be a facet mismatch for the same query.

There are many ways of computing similarities between products, and an advantage of PSL is that it supports the use of multiple similarity functions. Here, we make use of a latent product representations v_p for a product p, and then use cosine similarity to form the rules. Product representations are created by averaging word embeddings of the title words, the latter of which is learned using word2vec [27]. A similarity predicate can be created using the cosine distance between two vectors, i.e., $\mathit{Similar}(p_1, p_2) = \frac{\langle v_{p_1}, v_{p_2} \rangle}{\|v_{p_1}\| \|v_{p_2}\|}$. Another common method for defining product similarities is via collaborative filtering, where co-purchased or viewed items can be seen to be similar to each other. We also tried rules that make use of collaborative filtering in our model but did not see any improvements. The following rules are used to perform the collective inference on the *Mismatch* predicate:

$$\mathit{Mismatch}(Q, P_1) \wedge \mathit{Similar}(P_1, P_2) \rightarrow \mathit{Mismatch}(Q, P_2) \tag{7}$$

$$\neg\mathit{Mismatch}(Q, P_1) \wedge \mathit{Similar}(P_1, P_2) \rightarrow \neg\mathit{Mismatch}(Q, P_2) \tag{8}$$

By combining the above rules with rules (5) and (6) we can generate an HL-MRF which incorporates micrographs to perform joint predictions. We refer to this model as SMC.

5.3 Incorporating Confidences into Mismatch Detection

Propagating the right information in the micrograph is key to improving the predictions of the model. Specifically, we want to be able to boost the performance on (query, product) pairs where the TMC cannot confidently predict whether there exists a facet mismatch by propagating information from other (query, product) pairs where the TMC has high confidence. To this end, we introduce two new predicates called *StrongMismatch* and *Strong*TMC. A *Strong*TMC prediction is one where the TMC score is above (below) a prespecified threshold indicating a match (mismatch). $\mathit{Strong}\text{TMC}(Q, P)$ exists for all (query, product) pair for which the $\text{TMC}(Q, P) > lim_U$ or $\text{TMC}(Q, P) < lim_L$ and $lim_U, lim_L \in [0, 1]$. To use these strong predictors we introduce two more rules:

$$\mathit{Strong}\text{TMC}(Q, P) \rightarrow \mathit{StrongMismatch}(Q, P) \tag{9}$$

$$\neg\mathit{Strong}\text{TMC}(Q, P) \rightarrow \neg\mathit{StrongMismatch}(Q, P) \tag{10}$$

$\mathit{StrongMismatch}(Q, P)$ is a target predicate to infer. The above rules tell our model to "trust" the TMC when the latter is confident in its predictions.

It is important to note that the above rules are different from the rules in the previous subsection (like (7) and (8)). Specifically, the former rules incorporates all scores generated by the TMC using behavioural and lexical features. While the rules (9) and (10) encode an amount of "trust" in the underlying TMC: the query-product pairs for which the confidence in the classification is high can be used as an important signal to incorporate the structure.

The predictions made using TMC are usually good on the products with strong behavioral data associated with them. However, such information is absent on a majority of items, either due to lack of

user signals or bad product curation. The primary idea is to improve the predictions on these products by propagating information from similar products where there is strong behavioral data.

Therefore, we propagate only the strong predictions only on micrographs that have them, i.e., on a filtered set of queries. We consider *Strong*TMC scores and *StrongMismatch* for only those queries that contain at least one product with a strong TMC prediction score and one product without a strong TMC score. The mismatch values for other queries are derived directly from TMC scores and are not altered. We refer to this approach of propagating only the high confidence score as *strong* SMC (s^2MC).

This targets queries for which there are products whose classification can be improved by propagating scores. This can be encoded using the following rules:

$$StrongMismatch(Q, P_1) \wedge Similar(P_1, P_2) \rightarrow StrongMismatch(Q, P_2) \quad (11)$$

$$\neg StrongMismatch(Q, P_1) \wedge Similar(P_1, P_2) \rightarrow \neg StrongMismatch(Q, P_2) \quad (12)$$

The above rule states that only a strong prediction from TMC will be propagated to similar products. Further, this information can be propagated to (*Mismatch*) using the following rules:

$$StrongMismatch(Q, P) \rightarrow Mismatch(Q, P) \quad (13)$$

$$\neg StrongMismatch(Q, P) \rightarrow \neg Mismatch(Q, P) \quad (14)$$

Eventually after performing inference if $Mismatch(Q, P) > t$ then the query-product pair is considered a mismatch.

5.4 Using prior as a regularizer

A prior rule is usually used to regularize the values. A negative prior is usually placed on the target predicates *Mismatch* and *StongMismatch*. This is encoded in the following manner:

$$\neg Mismatch(Q, P) \quad (15)$$

The prior acts exactly like the priors in the Bayesian inference literature. The negation in the prior is reasonable: since we expect the majority of (query, product) pairs to be a match. Further, a negative prior can also be seen as a L2 regularizer used in statistical machine learning [3].

For our model, we use all the rules described so far. The weights for each of these rules are learned through grid search. More details about the learned weights are mentioned in Section 7.2. Further, we use squared hinge-loss potentials for all our rules. To solve the HL-MRF generated we derive and use a quasi-Newton (convex) optimization approach discussed in the next section.

Note that the final output of the model we build will be Mismatch(Q, P). The rules we define are to make sure that the value of Mismatch(Q, P) is accurate, and ideally better than that returned by the baseline TMC.

We round off this section with a representative example to explain the key idea. Consider the query q to be "black apple iphone", a popular product (p_q^1) to be "black iphone case" which is falsely associated with q, and a new product (p_q^2) to be "golden iphone case", which is also falsely associated with the query. The objective is to predict γ_{q,p_q^1}, and γ_{q,p_q^2} correctly as a facet mismatch. We train a TMC that makes predictions on the facet mismatch values. Let $\rho_{q,p_q^1} = 1.0$ and $\rho_{q,p_q^2} = 0.5$. In our approach, using rules (11) and (12), we improve the γ_{q,p_q^2} by propagating ρ_{q,p_q^1} and jointly inferring the values γ_{q,p_q^1} and γ_{q,p_q^2}. This results in $\gamma_{q,p_q^1} = \gamma_{q_1,p_q^2} = 1.0$.

6 SCALABILITY

We want our proposed method to perform efficient inference at web scale (i.e., each micrograph inference takes only a few milliseconds or less). Typically, inference in HL-MRFs are solved using ADMM. Although ADMM is scalable and can handle large datasets, it is not fast enough in terms of convergence to meet the stringent latency constraints of an e-commerce website. Of the many optimization methods second order (Newton) methods are known to have the fastest convergence. However, their per iteration cost increases as the size of data increases. Since we deal with micrographs, for the inference problem in (4) each micrograph can be solved independently. This implies that the instantiated model is small and it becomes feasible to use a quasi-Newton method. Specifically, we use the trust-region Newton method (TRON) [16]. In this section, we show that the optimization problem (4) is similar to that of (squared) SVMs, which in turn enables us to use solvers based on TRON [11].

To be able to use a quasi-Newton method we need to ensure that our objective is strongly convex, however, (4) is not. We thus add an L2 regularizer to the objective. The new objective function can be written as:

$$f(\mathbf{Y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{Y}, \mathbf{X}) + \lambda ||\mathbf{Y}||_2^2 \quad (16)$$

$$\underset{\mathbf{Y}}{\operatorname{argmax}} P(\mathbf{Y}|\mathbf{X}) = \underset{\mathbf{Y}}{\operatorname{argmin}} f(\mathbf{Y})$$

$$s.t. 0 \leq y \leq 1, \forall y \in \mathbf{Y}$$

$$\phi_i(\mathbf{Y}, \mathbf{X}) = max(l_i(\mathbf{Y}, \mathbf{X}), 0)^2$$

where λ is a hyperparameter. Note that the regularizer effectively replaces the prior rule for the model [3], and so the value for λ can be the same value as the weight of the prior rule (15). We can write the term l_i in the potential function ϕ_i as the following:

$$l_i(\mathbf{Y}, \mathbf{X}) = \mathbf{Y}^T \mathbf{z}_i + c_i \quad (17)$$

where $\mathbf{z}_i \in \{1, -1\}^m$ is a vector that indicates which unobserved random variables participate in the potential i, m is the total number of unobserved random variables, and $c_i = \mathbf{X}^T \tilde{\mathbf{z}}_i$, where $\tilde{\mathbf{z}}_i \in \{1, -1\}^{\tilde{m}}$ is a vector that indicates which observed random variables participate in the potential i, $\tilde{m}$ is the total number of observed random variables. The first derivative for the new objective can be written as:

$$\frac{\delta f(\mathbf{Y})}{\delta \mathbf{Y}} = 2\lambda \mathbf{Y} + \sum_{i \in S} 2 w_i \mathbf{z}_i (\mathbf{Y}^T \mathbf{z}_i + c_i) \quad (18)$$

where $S := \{i \in 1, 2, \ldots m, \ \ |\mathbf{Y}^T \mathbf{z}_i > -c_i\}$. The (generalized) Hessian is given by,

$$\frac{\delta^2 f(\mathbf{Y})}{\delta \mathbf{Y}^2} = 2\lambda \mathcal{I} + \sum_{i \in S} 2 w_i z_i z_i^T. \quad (19)$$

Using the first and second derivative (18 and 19) we can use quasi-Newton methods to perform inference. This has two distinct advantages over first order methods like ADMM:

- The number of iterations needed to converge to the optimal value is often orders of magnitude lower.
- Each iteration of the ADMM method requires solving a set of linear equations to convergence. In contrast, we only take a few steps of conjugate gradient method to obtain an approximate solution. The approximate solution coupled with the second order updates has shown to be highly successful in practice [14]. This significantly reduces the per iteration cost of the second order method.

We make use of the liblinear package [11]. The SVM objective obtained for L2 regularized L2-loss in the primal form is very similar to the Equation 17. Specifically, from (17) we see that the c_i play the role of a data-specific margin, while the "labels" for each point can be seen to be -1. Concretely, we can rewrite (17) in the squared SVM form as,

$$f(Y) = \sum_{i=1}^{n} w_i max(0, c_i - (-1)Y^T(z_i))^2 + \lambda \|Y\|^2. \quad (20)$$

The only exception is the extra box constraint on $\mathbf{Y}$ which can be easily enforced [23].

7 EMPIRICAL EVALUATION

In this section, we show the power of using micrographs to improve facet mismatch classification. We use multiple sampled datasets to evaluate our approach. First, we show that the TRON based method we proposed (20) is up to 150x faster than the baseline ADMM solver for the PSL, more specifically for facet mismatch classification. We show that the rules we described earlier are indeed useful and the performance of the classifier suffers when we omit these rules. Finally, we compare and contrast various methods and show that our proposed approach significantly outperforms the baseline methods.

7.1 Datasets and Models

To evaluate our approach we use three datasets from a popular product listing website where the user types a query and sees a ranked list of products. We refer to these datasets as D1, D2, and D3. The datasets correspond to (query, product) pairs shown to the users, with other features obtained from search logs. The most egregious form of facet mismatch is along the product type facet (compared to brand, color, size, etc.) and we focus on that in this paper. We used human annotators to obtain the ground truth data for all our (query, product) pairs. If a (query, product) pair does not match along the product type facet, the judge marks the pair as mismatched. For example ("iphone x", "iphone x case") is a product type mismatch and ("iphone x", "iphone x refurbished") is a product type match. We have listed the dataset details in Table 1. We use the human annotated labels to perform evaluation only, i.e., we consider all variables generated by the *Mismatch* and the *StrongMismatch* predicates as unobserved. We use only aggregated and de-identified information for our experiments (i.e., they do not include personally identifying information about individuals in the dataset).

Table 1: Details for the three datasets we use. Even though the dataset contains small number of queries, the query independent nature of our approach enables our results to hold for even larger datasets.

Dataset	Queries	Products
D1	1194	7790
D2	149	866
D3	591	1959

Table 2: Different models and rules used to perform evaluation.

Model name	Rules used (Equation number)	Weights for rules
TMC	5 and 6	1 and 1
STMC	5, 6, and 15	1000, 1000, and 1
SMC	5, 6, 7, 8, and 15	100, 100, 10, 10 and 1
S^2MC	5, 6, 11, 12, 9, 10, 13, 14, and 15	10, 10, 10, 10, 1000, 1000, 100, 100 and 1

As mentioned earlier, we used TMC as our baseline model. Each training data point is a (query, product) pair and the features included were lexical (text similarity) and behavioral (likelihood of click, add, and purchase). The target (or dependent variable) for this model is binary indicating whether a specific (query, product) pair is a facet mismatch or not. The TMC model was trained on a human annotated dataset with $\sim 200K$ query-product pairs. We compare this model to the SMC models proposed in this paper. The first model defined through PSL adds a prior to the TMC score to regularize/smooth the values (using the prior rule), we refer to this model as *smoothed TMC* (STMC). The next PSL model defined uses the micrograph structure defined through product similarities to propagate the TMC scores and perform a joint prediction on mismatches. We refer to this model as *SMC*. The final model propagates TMC scores only from high confidence edges through product similarities and perform a joint prediction on mismatches, which we refer to as S^2MC. We describe the rules used by the models and their weights in Table 2.

The weights mentioned here were obtained by performing a grid search over a set of weight options using a train dataset. To compute the similarities between products, we train a word2vec model using about 30M product titles from the catalog of a popular e-commerce website. Each product embedding is the average of the word embeddings in the title (at first glance this might look naive, but for our specific dataset title contains plenty of information and proved sufficient enough).

We need to quantify the strength of strong mismatch using an upper and lower limit on the TMC scores. Specifically, assume that the classifier predicts the probability of a (query, product) pair being a match. Then, we need a threshold lim_U so that any score above lim_U is a strong match and any score under a threshold lim_L is a strong mismatch. At the limit point when $lim_U = lim_L$, all TMC scores are classified as strong predictions. While this may increase the coverage of queries that have strong predictions, it

Table 3: Number of queries that uses micrograph (Coverage) for different lower and upper limit.

lim_L	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10	0.11	0.12	0.13	0.14	0.15
lim_U	0.94	0.88	0.82	0.76	0.70	0.64	0.58	0.52	0.47	0.40	0.35	0.30	0.25	0.20	0.15
Coverage	0.96	0.90	0.83	0.78	0.73	0.69	0.64	0.60	0.58	0.54	0.50	0.43	0.36	0.24	0.00

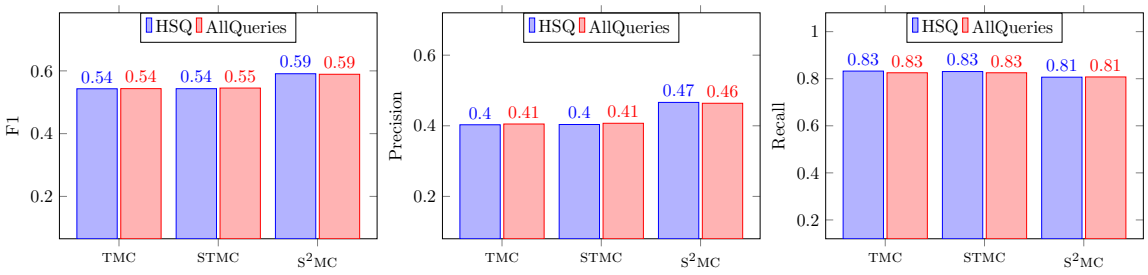


Figure 2: This figure shows the comparison of three models using TMC vs. STMC vs. s^2MC on D1 with HSQ coverage 60%.

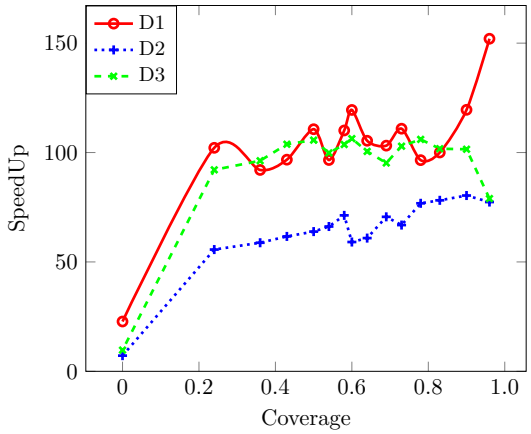


Figure 3: Speedup obtained by using TRON over ADMM at performing inference. The speedup increases as coverage increases and we get a speedup of up to 150X on the D1 dataset.

also decreases the quality of the scores used for propagation. At this limit point, including strong mismatch is the same as the model defined using *SMC*. Therefore, we choose to use strong predictions to only those queries that contain at least one product with strong prediction and one with weak prediction. The intuition being, a strong prediction can be propagated to a weak prediction if the two products are similar to improve the overall quality of the predictions.

We denote the set of *High Scoring Queries* (HSQ) as queries that are covered by the strong mismatch rules. In the sequel, the suffix 'HSQ' on a model indicates the score obtained by considering only such queries. Lack of a suffix means that we evaluate the performance of the entire dataset. A key thing to note is that the s^2MC model uses micrographs for only the HSQs, whereas SMC uses micrograph on all queries.

As $lim_L \rightarrow lim_U$, the number of HSQs will approach 0. We refer to the fraction of HSQs and total queries as the "coverage", as these will be the queries that make use of micrograph to predict facet mismatchusing the s^2MC model. We show the different values used for lim_U and lim_L and the corresponding coverage in Table 3. Further, we use the PSL open source code[1] to perform inference.

7.2 Experimental setup and evaluation

7.2.1 *SPEEDUP FROM USING TRON.*

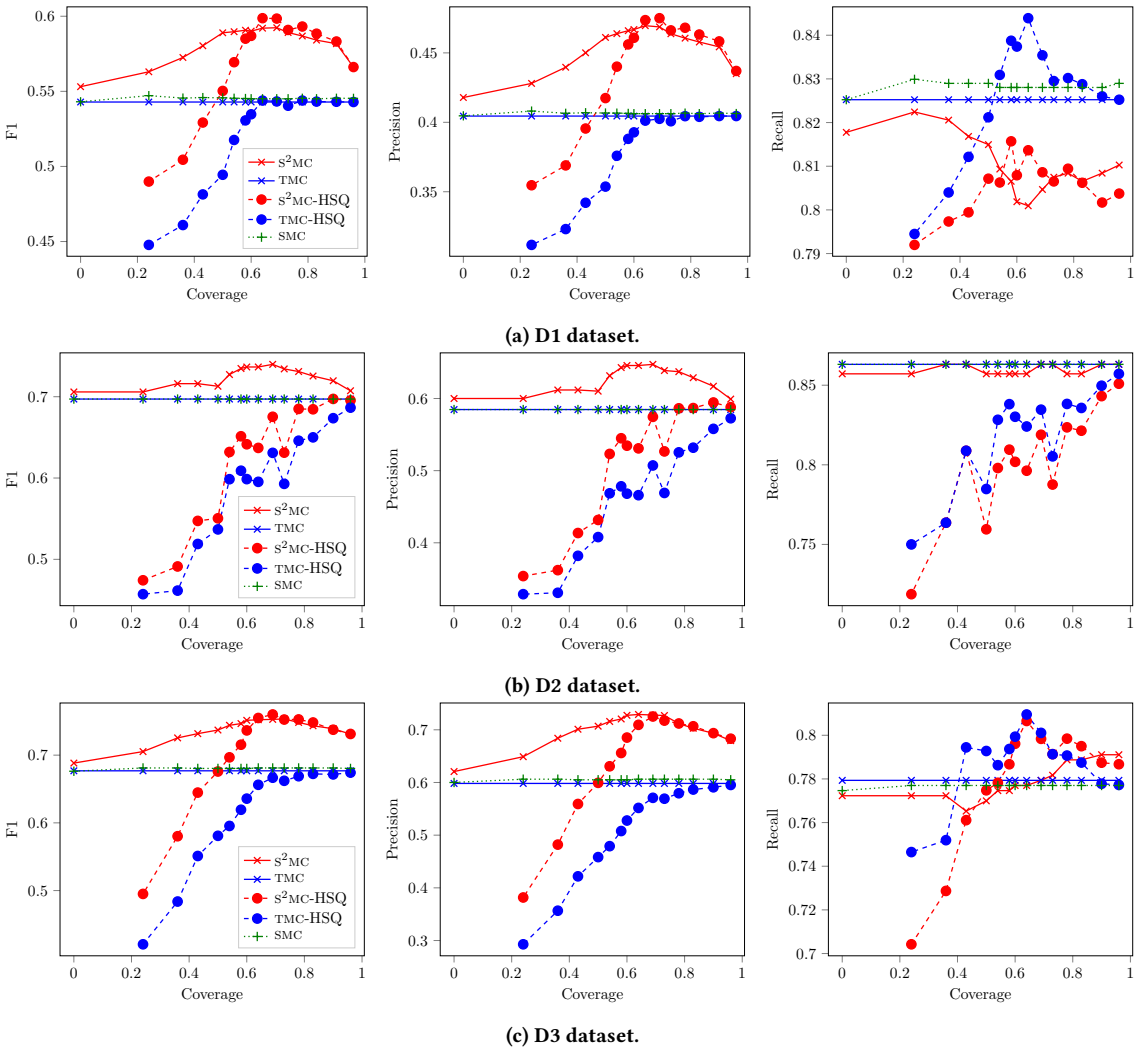
We show that using TRON makes the s^2MC run significantly faster than ADMM. We run inference using both TRON and ADMM for all queries in each dataset and report the speedup obtained. To perform inference using TRON, we make minor changes to the open source liblinear package [11] to adapt to the HL-MRF objective. To have a fair comparison, we re-implemented ADMM for HL-MRF in C++. We report the result of this experiment in Figure 3. As we increase the coverage TRON is up to 150X faster than ADMM. The speedup obtained is minimal when the coverage is 0, i.e., when we used only TMC's, and there are no micrographs that slow down ADMM. As we start covering more queries, the speedup also increases. This demonstrates why TRON methods are powerful for online inference when using micrographs, particularly, when we use s^2MC. Further, we also observe that when using TRON, per-query predictions time averages to about 0.1 milliseconds.

7.2.2 *IMPACT OF MICROGRAPHS.*

The next question we want to answer is how much do micrographs actually help, over and above the TMC predictions. Specifically, we consider three cases for the dataset D1: using the TMC model, including the prior (rule (15)) which we refer STMC, and s^2MC. We use the D1 dataset with HSQ coverage 60% [2]. Figure 2 shows the result obtained from this experiment. We observe that when TMC is smoothed with some amount of regularization, there is no improvement in the metrics. However, as we add the information from the micrographs by using s^2MC, we see a significant boost. We notice about 7% increase in precision, 6% increase in F1. We also observe similar boost in HSQs. TMC scores tend to be polar for products with high behavioural information. Therefore, the improvement on

[1] http://psl.linqs.org/

[2] We will show in the sequel that this coverage value performs the best



(a) D1 dataset.

(b) D2 dataset.

(c) D3 dataset.

Figure 4: This figure shows precision, recall and F1 for TMC, STMC, SMC, and s²MC on three different datasets, D1 (top), D2 (middle) and D3 (bottom). We show the metrics for both HSQs and all queries included. Optimal performance is obtained when s²MC used with threshold of 0.08 and 0.52 is used as lower and upper limit resulting in s²MC model affecting 60% of queries.

HSQs indicates that the predictions on products with lesser user interactions (likely tail products) have been improved.

7.2.3 COMPARISONS WITH MULTIPLE BASELINES. Finally, we compare TMC, STMC, SMC, and s²MC as described in Table 2 over multiple coverage values. Note that when coverage is 0, s²MC uses no micrograph and alternatively, when coverage is 1 s²MC uses micrograph for all the queries. We intend to find out if there's a certain coverage value that maximizes performance. Note that a higher coverage need not necessarily translate to higher performance, since the underlying TMC model might have noisy predictions, leading to noisy edges in the micrographs, in turn leading to incorrect final predictions.

We report our findings in Figure 4. Here we observe that in all datasets we see a clear improvement of precision and F1 in both the HSQ and all queries, while keeping the recall relatively constant.. We observe up to 7% increase in precision for all queries in D1 dataset and about 6% increase in F1. We see a similar trend in other datasets concerning precision and F1 with a maximum boost of 12% in precision in D2. However, we do see a relatively small drop of 2% in recall in the D1 dataset and no drop in recall values in other two datasets.

We also notice that as expected, the evaluation metrics for HSQ in TMC is always lower than the overall value. Since HSQ contains at least one product for which the TMC prediction score was not high. This implies that there is at least one product for the query which is more likely to be misclassified by TMC predictions alone. Therefore, the overall HSQs are likely to have relatively more incorrect TMC predictions than the average of all queries.

We also observe that the dataset generated using $lim_U = 0.52$ and $lim_L = 0.08$ resulting in 60% coverage yields the maximum improvement in both precision and F1. Another thing to note is that we observe the metric values obtained using SMC are almost the same as that of TMC. Finally, we see that when micrographs are not used at all (coverage = 0), the metric values are almost the same. The slight difference is due to the regularization of the TMC predictions and noise.

8 CONCLUSION AND FUTURE WORK

We showed in this paper that structural information can be used to improve facet mismatch classification in modern e-commerce search engines. We introduced the concept of a micrograph, that can be used to incorporate additional structure between queries and products, and reduced the problem to an inference of a graphical model using PSL. The methods we proposed yield impressive gains over baseline methods. We also re-cast the problem with a strongly convex objective, allowing us to use scalable second order approaches and make the inference viable to real time vending of search results. Through experiments, we show that our approach achieves 150X speedup over existing solvers and up to 6% improvement in terms of F1 score.

We hypothesize that incorporating the (query, query) edges in our micrograph will improve the effectiveness of our approach even further. However, keeping in mind the latency constraints of our approach, we have not included them in this paper. As part of the future work, we would like to explore incorporating this additional information, while not loosing on the latency requirements.

9 ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation grants CCF-1740850 and IIS-1703331, AFRL and the Defense Advanced Research Projects Agency, and Amazon LLC.

REFERENCES

[1] Charu C. Aggarwal. 2014. *Data Classification: Algorithms and Applications.* Chapman & Hall/CRC.
[2] Sanjay Agrawal, Kaushik Chakrabarti, Surajit Chaudhuri, Venkatesh Ganti, Arnd Christian Konig, and Dong Xin. 2009. Exploiting Web Search Engines to Search Structured Databases. In *Proceedings of the 18th International Conference on World Wide Web.*
[3] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research* 18 (2017), 109:1–109:67.
[4] Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction. In *UAI.*
[5] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag.
[6] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* (2011).
[7] Michael J. Cafarella, Michele Banko, and Oren Etzioni. 2006. Relational Web Search. In *Proceedings of the 15th International Conference on World Wide Web.*
[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*
[9] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole. 2016. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation.* Morgan & Claypool.
[10] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. 2017. ZooBP: Belief Propagation for Heterogeneous Networks. *Proc. VLDB Endow.* 10 (2017), 625–636.
[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.* 9 (2008), 1871–1874.
[12] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* (2000).
[13] Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning.* MIT.
[14] Siddharth Gopal and Yiming Yang. 2013. Distributed training of Large-scale Logistic models. In *Proceedings of the 30th International Conference on Machine Learning.*
[15] Benjamin Haeffele, Eric Young, and Rene Vidal. 2014. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *International Conference on Machine Learning.* 2007–2015.
[16] Chih-Yang Hsia, Ya Zhu, and Chih-Jen Lin. 2017. A Study on Trust Region Update Rules in Newton Methods for Large-scale Linear Classification. In *Proceedings of the Ninth Asian Conference on Machine Learning*, Vol. 77. 33–48.
[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems.*
[18] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A Short Introduction to Probabilistic Soft Logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications.*
[19] George J. Klir and Bo Yuan. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Prentice-Hall, Inc.
[20] Xiangnan Kong, Philip S. Yu, Ying Ding, and David J. Wild. 2012. Meta Path-based Collective Classification in Heterogeneous Information Networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management.*
[21] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. 2014. Aggregated Search: A New Information Retrieval Paradigm. *ACM Comput. Surv.* 46, 3 (2014), 41:1–41:31.
[22] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems. In *9th ACM Conference on Recommender Systems (RecSys).* ACM, ACM.
[23] Chih-Jen Lin and Jorge J. Moré. 1999. Newton's Method for Large Bound-Constrained Optimization Problems. *SIAM J. on Optimization* (1999).
[24] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. 2008. Trust region newton method for logistic regression. *Journal of Machine Learning Research* 9, Apr (2008), 627–650.
[25] Nickel Maxmilien, Murphy Kevin, Tresp Volker, and Gabrilovich Evgeniy. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.
[26] Rada F. Mihalcea and Dragomir R. Radev. 2011. *Graph-based Natural Language Processing and Information Retrieval.* Cambridge University Press.
[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26.*
[28] Houssam Nassif, Yirong Wu, David Page, and Elizabeth S. Burnside. 2012. Logical Differential Prediction Bayes Net, improving breast cancer diagnosis for older women. In *AMIA.*
[29] Singla Parag and Domingos Pedros. 2006. Entity Resolution with Markov Logic. In *Sixth International Conference on Data Mining.*
[30] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems.* 2107–2115.
[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29 (2008), 93–106.
[32] Dhanya Sridhar, Shobeir Fakhraei, and Lise Getoor. 2016. A probabilistic approach for collective similarity-based drug-drug interaction prediction. *Bioinformatics* 32, 20 (2016), 3175–3182.
[33] Charles Sutton and Andrew McCallum. 2012. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning* 4 (2012), 267–373.
[34] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019. Heterogeneous Graph Attention Network. In *Proceedings of the 30th International Conference on the web conference.*
[35] Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving Web Search Results Using Affinity Graph. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*
[36] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi, and Yangyong Zhu. 2018. Deep Collective Classification in Heterogeneous Information Networks. In *Proceedings of the 2018 World Wide Web Conference.*
[37] Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving Diversity in Ranking using Absorbing Random Walks. In *The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference.*