**A Contribution Toward a Text Matching AI-API**

Intern Name

DSBA 6400 | Fall 20xx

Mentor: First Last, Company name

Faculty Advisor: Professor Doug Hague

Table of Contents

[Executive Summary 3](#_Toc59222159)

[Introduction 4](#_Toc59222160)

[Methods 7](#_Toc59222161)

[The Ground Truth 7](#_Toc59222162)

[Machine Learning Model 8](#_Toc59222163)

[Model Evaluation 9](#_Toc59222164)

[Results 11](#_Toc59222165)

[Discussion and Conclusions 15](#_Toc59222166)

[Appendix 16](#_Toc59222167)

Executive Summary

Company A was sought to provide a solution to the common yet difficult challenge of string matching. A Midwest software company was struggling with a pervasive challenge of scaling operations in a competitive landscape of online retailers, where manufacturers need to track their products across a vast ecosystem of platforms. Manufacturers need to efficiently track their products and ensure partnership agreements are upheld, one important agreement is Minimum Advertised Price (MAP). An intern was needed for pertinent project tasks. This project highlights a deficiency in current string match offerings. Current matching strategies attempt to address various matching constraints but are not scalable for this application. The project is dynamic, but a robust classification framework was confidently built and tested. A curated ground truth was established for optimal development and a final model version achieved 0.907 AUC and 5.414 log loss scores. Machine learning extracted insights about the importance of matching ratio where higher ratios provide more information on correct matches. A visual review function of model performance shows optimism for further developments. The model is prepared for next iterations and incorporation into an AI-API that will iteratively improve matching with exposure to new match examples. The success of this project means improving a company’s ability to scale with online sales channels while also protecting vital business aspects in branding and revenues.

Introduction

Online marketplace platforms offer a competitive landscape for manufacturers, distributors, and retailers alike. Amazon captures 50% of the total US e-commerce marketplace and remaining portions are distributed across a fractured setting of resellers, direct-to-consumer options, and brick-and-click retailers. Manufacturers must track their products and pricing across this vast domain to ensure their retailers abide by Minimum Advertised Price (MAP) policies. MAP policies define the price floor by which a product can be marketed and sold, it serves to protect profit margins, brand integrity, and relationships with trusted partners. Violating pricing standards is beneficial for retailers in the online arena because competitive pricing drives consumerism. Additionally, the explosive growth in online shopping is attracting more retailers to this platform. These trends force manufacturers to adapt in unprecedented scale to ensure MAP adherence.

In order for a manufacturer to efficiently find MAP violations, it must first develop an effective strategy of extracting products across the internet and matching them to their internal portfolio. Although topically straightforward, the execution of such a task is considerably arduous. Online vendors circumvent tracking by avoiding the use of manufacturer identifiers, forcing manufacturers to track their products using alternative product information. Company A was sought to solve such a challenge within a core business function of a Midwest software company. The project goal was to develop a solution that would improve matching between sources, thereby reducing operational costs, and enabling the company to scale its offerings.

The software company is hired by manufacturers to ensure retailers are abiding by MAP policies. They use the internal portfolio to scrape online products and retain matches that meet a threshold for certain information (e.g. product code, full title, partial title). Positive matches were stored and manually reviewed by employees, then evaluated for MAP violations. The current workflow performed poorly and generated considerable cloud compute and storage costs.

The solution needed to improve a critical point in their process workflow, increasing match accuracy. Improving match accuracy required redefining a positive match and providing strong negative match examples. The company was using manual overrides to keep the system working. This presents a use-case for machine learning because a model can learn to ignore data yet still successfully match. Capturing precise matches also addressed exorbitant cloud expenses by extracting more value and offsetting the cost of resources (i.e. less compute time spent evaluating candidates, and less storage consumed by useless matches with low informative power). Online vendors avoid using the same manufacturer SKU or Universal Product Code (UPC) in order to escape tracking. In the absence of identifiers, matching string / text information becomes a necessary alternative. The analytic approach focuses on generating reliable matches between the manufacturer portfolio and online contenders using product title. Challenges in string matching extend beyond string-to-string comparison and are not straightforward.

Text matching is deceptively complex. It is difficult to convey tacit human knowledge to a machine. For example, a human reader can equate “*Women’s sm blue fleece jacket*” to “*Fleece jacket (color: blue, size: small, gender: women)*”. The challenge is more obvious when comparing strings that contain units of size or measure; “*inches*” can be represented as the word, by a double quote following a number (*12”*) or abbreviated as “*in.*” Furthermore, comparing strings that contain ambiguous information is an additional hindrance, does “*blk*” mean “*bulk*” or “*black*”? What about matching instances that contain ranges (e.g. *Men’s Mizuno Wave Rider size 9-12*)? How should one convey the importance of word order (e.g. *KitchenAid KM74055 Stand-mixer* | *KM74055a Ruby Red KitchenAid Stand mixer*). There is unbounded potential within categories, product lines, or brands. Thus, a heuristics approach is unreasonable, it would require encoding every possible word representation (i.e. “inch” = “in” = #”) and be difficult to maintain at scale.

Existing matching techniques are inadequate for this application. Methods like fuzzy string matching, word vectors, or distance-based techniques are challenged by the ambiguity that exists within product titles. Fuzzy matching is an approximate technique that leverages Levenshtein distance to compare strings. This strategy is naively sensitive to minor differences and disqualifies matches with missing, abbreviated, or extra words, which are omnipresent characteristics of product titles. Vectorization methods lean on dictionaries of words and associated numerical value, and even significant training data does not lend itself to meaningful vector learnings. A system would need to store a numerical representation for “*b, bl, bk, blk, blc, black,*” that might represent the word “*black”*, but it could not store a different vector when “*blk”* represents *“bulk”*. Vector weights are typically used in general English language, but the meaning of “*litewave*” for The North Face will not have meaning beyond a brand. These critical challenges justify a token-based matching strategy.

Certain data science techniques, including machine learning, can address the aforementioned complications. An optimal matching solution should allow for scalability, improved accuracy, efficient use of resources, and reduced human input. Company A proposed an approach that avoids branching techniques that can’t generalize; evaluates the string characteristics using metrics that quantify the individual strings and the relationship between the strings; leans on machine learning to provide the insights about the most informative metrics for matching; and offers the framework as an AI-API whereby model improvement is gained with experience. The partnership between machine learning and the API is imperative; calls and information returns are essential to the learning process and remove the need for unscalable, arcane heuristics.

There were four business objectives for the internship. Task requirements began with establishing the ground truth where information was collected on 1,321 portfolio products. After settling the ground truth, I was required to build and tune a machine learning classifier that made inferences on matches. Significant attention was placed on selecting appropriate quantitative and qualitative performance metrics. Given the data, problem context, and business application, feature importance, area under the curve (AUC), log loss, and a calibration plot were used for model assessment. The final internship task was to write a match review function that enabled qualitative review of model success and failure at the individual product level.

Methods

The following methods were used to achieve each internship objective:

* *Develop the ground truth* – scrape information on internal product candidates, compare to external candidates, manually verify each positive match
* *Build and tune a logistic classifier* – partition the dataset, apply k-fold cross validation, develop in Python Jupyter notebook environment, evaluate and improve
* *Research performance metrics* – develop framework for assessing match inferences, apply metrics, iteratively improve the model
* *Code a manual match review function* – write a review function in Python that joins all inferences and raw data to review details on internal title, external title, class probabilities and predicted class labels

## The Ground Truth

The first internship objective was developing the ground truth, which contained two critical sub-tasks, create a positive matches and negative matches dataset. Positive matches must be manually verified for optimal information comparison and negative matches needed to provide a challenge for the classification model. In order to speed-up development time and throughput, three manufacturers were selected for a pilot model, The North Face, KitchenAid, and Logitech. In total, 1,321 products referred, to as ‘internal’, products were used for development of the ground truth.

The internal products were evaluated against 12,735 external match candidates scraped from Amazon. A one-to-many relationship meant each internal product could be matched to numerous external products and each external product could match several internal products. After tedious review, 2,692 positive matches were inputs for the machine learning model. As the intern, I scraped all internal products, used a fuzzy string-matching library to reduce the search space for matching, and manually confirmed each positive match. The mentor’s role in ground truth was collecting positive and negative match candidates from Amazon. I constructed a SQLite database in DB Browser from the ground truth data. Development was conducted in a Jupyter notebook which was integrated with SQLite for fast querying and flexibility. I queried the tables directly from the Jupyter notebook, captured the query output in a Pandas data frame, and downstream processed with traditional data science Python libraries.

## Machine Learning Model

I built a baseline logistic classifier in Python using the scikit-learn library. Coding the model build and tuning were key functions required of the internship. Model versions were tracked in a shared folder within Google drive along with raw and transformed data. My mentor coded the initial feature library (string and match metrics) which quantified the strings and string-relationships.

One of the first pre-modeling considerations was sampling strategy. The data was considerably skewed, the majority class (not-match) overwhelmed the minority (match) class 19:1. Early model versions had unrealistic AUC and log loss metrics indicating false model confidence that was likely due to unchallenging negative matches (not-matchers were *easy* not-matches). The first solution was down sampling the majority class to a 3:1 ratio. A moderate improvement in model outcome was observed.

The next iteration focused on removing dependency on length metrics. The model was relying on common patterns, such as internal product titles being shorter than external product titles, and incorrectly labeling the examples as not-match. In efforts to correct this bias, I revisited the raw data and filtered for title length to give matches a better advantage and increase the challenge for the classifier. Only rows with an external title less than two standard deviations from the length of the internal title were retained. This strategy naturally balanced the classes and benefited downstream work; therefore, down sampling majority class was no longer necessary.

The final classifier was trained on a 90/10 split with 3-fold cross validation. The entire dataset contained the following target distribution: 4,717 not-match and 2,172 match observations. Inputs were standardized and imputed when missing using sklearn libraries (Simple Imputer, Standard Scaler). To some disadvantage the was model fit on four features (with more pending development) and yielded optimistic insights for further development. Performance was assessed using four quantitative methods and one qualitative.

Logistic regression is appropriate for this application because the target is binary (match or not) and the probabilistic output means I can evaluate uncertainty associated with predictions. Each observation is assigned a probability for belonging to one of two classes. The predicted label is assigned given the class with the highest predicted probability. Probabilities from a logistic classifier gives insights into model confidence, which we can exploit to improve inferencing. For example, if the model incorrectly labels a given observation as not-match with 0.75 confidence, we can leverage the manual review function understand why the model was incorrectly confident and adjust upstream metrics to compensate for this weakness.

## Model Evaluation

The following four quantitative methods were used to evaluate model performance, feature importance, area under the curve (AUC), log loss, and a calibration curve. The final internship task was to code a custom match review function allowing developers to qualitatively review model performance. In this way, one could assess details of model success and failure.

Feature importance is significant in using machine learning for this project. It provides subjective insights into exact string metrics that help or impede matching. The following metrics were inputs to the classifier: length of title, ratio of words that match, ratio of average word length, words matched in order, and longest word length matched. These metrics quantify string characteristics.

AUC is an important metric for evaluating a logistic classifier because it presents a scale-invariant comparison of models. As an aggregate method of threshold-invariance, it provides details on separation strength between the classes. Unrealistically high AUC indicates an ill-fit model. The training data in this project was imbalanced and the initial model had an AUC 0.999 because it incorrectly labeled all training observations as the majority class. Log loss is less intuitive but is an industry standard evaluation of probabilistic models. It’s a measure of classification uncertainty where values closer to zero means the model is more confident (less uncertain). Early model versions had low log loss (0.002, 0.06 respectively) but the class imbalance and exceptionally easy training examples were the reason for such observations. Quasi-tuning was conducting by custom sampling the training data based on external title length, which increased the challenge for the matcher.

A probability calibration plot is a visual tool to assess confidence on prediction output from a probabilistic model. Most importantly, it provides details of where the model is succeeding and failing by binning the predicted class probabilities and plotting the average probability within each bin. A well-calibrated model will hug the optimized calibration line indicating that prediction confidence is increasing proportionally with increasing bins of average probability prediction.

**Match Review Function**

The review function randomly samples from the calibration bins and displays the internal product title, external Amazon product ID (ASIN), external product title, ground truth label, predicted label, and probabilities of belonging to each class. This allows for precise injection of machine learning because weaknesses in model inferences can be viewed in highest granularity.

**Additional Remarks on Methods**

The original project goal was to incorporate the matching framework into an AI-API. Unfortunately, data challenges prevented this. The ground truth required several revisions. Partly because unintended bias was introduced and required correction, but also due to early decisions about ground truth strategy. For example, product version was ignored in determining match status, *‘m eco nuptse vest’* was matched to ‘*mens nuptse vest’* but this caused problems during inferencing. The model assigned a confident probability (0.749), not a match. Therefore, bias from a beginning strategy influenced model outcomes. This is currently under review for correction.

Results

The following results are presented in the context of the business objectives proposed for the internship. All the tasks were successfully completed during the internship except for the development of an AI-API. There were a host of challenges around data readiness for production, selection of cloud environment offerings, and long-term model hosting options that prevented success of the AI-API within the internship timeframe. But, the AI-API is an important function of the solution which cannot be understated. The API is currently being developed for production using AWS Lambda functions in the cloud. It will eventually become a product on the AWS solutions marketplace.

Chart

Description automatically generatedThe most important feature for correctly matching products is the feature Exact Words Matched. Figure 1 displays the ranked coefficients from the most recent model version. The outcome is intuitive, when more words match between two product strings, they are more likely to be a correct match. It is interesting to see that Maximum Word Length has the smallest contribution toward matching, I assumed longer words would provide more information toward matching. The second most important feature is also intuitive, it captures word order relationships between product titles. If two product titles have more words matched *in order,* they are more likely to be a correct match. Thus far, the model upholds the reasons for using a token matching strategy. High-level token characteristics like Exact Words Matched and Words Matched in Order provide enough information for the model to classify the products.

Figure 1: Current model feature importance

Figure 2 displays the calibration curve for the current model version. Again, this visual is useful to review details on model performance. The top region of the calibration plot does Chart, line chart

Description automatically generatednot provide much insight, but the histogram (the bottom region) provides interesting information about model confidence. The histogram shows the majority of the inferences fall within 0.1 to 0.5, this grouping implies the model is not very confident in predicting matches in the current dataset. This is intentional, because the training examples were very difficult and reduced the model confidence from baseline.

Figure 2: Reliability of current classifier

As previously mentioned, log loss describes prediction uncertainty and this metric increased from 0.06 to 5.414 to the model presented here. Challenging the model is necessary because the matching framework will eventually be incorporating into the API where the ground truth will be refreshed. Therefore, the model should be adequate in detecting string details without overfitting the training data or underfitting new examples. Meaning, it will learn salient information that helps match and ignore information that does not. If we provide training data that is too easy, the model learns very little and will rely on undesirable characteristics like string length instead of realizing the importance of words matching in order between the strings. These considerations aim to achieve teaching the model generalizable lessons that can scale with the needs of the software company.

Testing the model and evaluating output is supported by manually reviewing instances of model success and failure. This is the qualitative piece of the solution. Figure 3 is sample output from the function. This function returns the internal portfolio title, the Amazon ID, the external title, ground truth label, predicted label, and probability of belonging to class 0 and 1.

The manual review function presents a critical view of model performance. This also speaks to the challenges with training data. Early decisions, like ignoring product version (as shown here), have downstream effects. The model is confident these products are not a match (Prob 0 = 0.745) but the ground truth is a match (Label = 1). The most important feature extracted earlier explains this behavior. The more words matched, the more likely a correct match. In this example, only three tokens are shared between the product titles and the model concluded this is not a match. We are still having conversations around solving this problem because it’s likely to affect other categories or brands.



Figure 3: Sample output of Manual Match Review

One of the most critical experiences I gained was in the modeling lifecycle. I learned the importance of data quality, and how to evaluate model efficacy with intentional metrics selection. Model lifecycle is more than a visual tool used to teach willing learners about a project workflow. The iteration between model development, data updates, and check in with decision makers is dynamic. In this project, one iteration would reveal a training data issue while the next would highlight feature interactions. The model lifecycle is only complete once harmony is established between model performance, data readiness, and actionable insights are extracted and accepted by stakeholders.

Several courses vitally contributed toward success in my project. Big Data Design (DSBA 6160) provided the foundational knowledge and applied experience in SQL. I constructed a SQLite database in DB Browser from the ground truth and queried tables within the database from the development environment. Advanced Business Analytics, ABA (DSBA 6211) provided extremely important lessons in data preprocessing, statistical modeling, and model evaluation. My internship project required the direct application of data normalization skills to the feature inputs in the logistic regression model. I also applied ABA course work in understanding the logistic classifier output; Dr. Zhao spent considerable time teaching how to build a logistic model, criticize the output, determine overfitting and underfitting, and tune accordingly. Applied Machine Learning, AML (DSBA 6156) was instrumental in my understanding of feature importance and engineering, building production-ready code frameworks, and evaluating ML model performance. There was no useful solution for extracting feature importance, I had to code one using the concepts covered in AML. The model will eventually reach a final version and become part of a larger pipeline – I had to code variables that would seamlessly integrate into a production environment. AML provided the necessary platform for learning this firsthand before implementing it into my project. Cloud Computing (DSBA 6190) was also necessary for understanding cloud implications, like the benefits of migrating the framework to the cloud, which cloud resources were viable, how to select cloud services given the problem context, and scenarios where the cloud would not provide added value. Even though the AI-API was not achievable within this project timeline, early decisions about how the project could reach production were heavily discussed. The topics I learned in cloud computing helped to conceptualize the cloud services and their necessity in the scope of my project.

All aforementioned courses emphasized the importance of adequate data in terms of quantity (number of observations/features, more data is often better) and quality (a lot of data is useless if considerable amount is missing). This cannot be understated. In this project experience, certain problems required revisiting the data multiple times, re-modeling, and observing any changes. This practice was the most authentic exposure I could gain in the beginnings of a data science project.

Discussion and Conclusions

Company A was pursued to correct a failing process within a core function of a software company. The company provides a critical service for manufacturers, assisting them with product tracking across online platforms and ensuring their retailers are upholding minimum advertised price standards. The company’s current processes required manual overrides in order to operate and was degrading with growth in the internal portfolio. Finding an adequate solution was burdened by challenges around data quality and availability. The solution is not fully complete but relevant impacts are apparent. First, the matching framework allows the software company to adequately provide a critical service for its manufacturing clients. Once the AI-API is deployed, Company A can provide real-time performance monitoring, refresh the ground truth to improve match inferences. This will propel a core business function for the company, reduce their operational costs, and open opportunity for vertically scaling their client portfolio. Downstream effects that manufacturers can better monitor MAP standards across online retailers, thereby protecting their profit margins, brand integrity, and maintain strong relationships with trusted partners.

Appendix

I am proud to have contributed toward a product that will become a functional service for an organization. A semester of work will ultimately become part of a solutions ecosystem for businesses across many industries. This internship was immense for my personal-professional development. It gave me experience in thinking about business problems through the lens of data science and the confidence to solve them. It also provided the challenge of solving such problems with actual business repercussions.

My internship experience was excellent. My mentor was diligent and readily available. He provided vague information and let me propose solutions and analyze the potential outcomes. He was also an invaluable asset who helped bridge the knowledge-to-practice gap between academics and industry. I greatly appreciate his guidance and profound project experience which he willingly shared. I was fortunate enough to listen during talks he gave to national business leaders and take part in developing his online course business leaders in data science. This exposure provided an inside look into data science in practice and the differences between academic and industry dialogues around data and analytical concepts.

All initial, empirical objectives were met during my internship. The ground truth was the most difficult hurdle to overcome, but diligence made the end product better. Prior coursework provided pertinent baseline knowledge for executing my ideas in model progress and performance assessment. My coding skills improved from the demands of manual functions, variable transformations, and aversion toward hard coding. I was also able to contextualize data science through the lens of business impact.