# Merchant Matching System: Technical Documentation

## Executive Summary

This document provides a comprehensive explanation of the hard-coded elements within our merchant matching system, which uses BERT MPNet alongside traditional string matching algorithms. Our approach balances computational efficiency with matching accuracy by strategically incorporating domain expertise through carefully calibrated weights and rules. Each hard-coded element serves a specific purpose in addressing the challenges of merchant name matching without requiring extensive labeled training data.

## 1. Introduction to Merchant Matching

Merchant matching is essential for financial data analysis, reconciling transactions across systems, and detecting fraudulent activities. The challenge lies in connecting different representations of the same merchant (e.g., “BofA,” “Bank of America,” “BANK OF AMERICA N.A.”). Our system implements a hybrid approach that combines semantic understanding (via BERT MPNet) with traditional string matching algorithms, enhanced by domain knowledge encoded through hard-coded weights and rules.

## 2. Base Weights in Similarity Algorithms

```
self.base_weights = {
    'jaro_winkler': 0.10,
    'damerau_levenshtein': 0.05,
    'tfidf_cosine': 0.05,
    'jaccard_bigram': 0.05,
    'soundex': 0.05,
    'token_sort_ratio': 0.10,
    'contains_ratio': 0.10,
    'fuzzy_levenshtein': 0.05,
    'trie_approximate': 0.10,
    'bert_similarity': 0.15,
    'aho_corasick': 0.05,
    'DBAName_formation': 0.15
}
```

### Why These Specific Values?

These weights represent expert knowledge about the relative effectiveness of each algorithm for merchant name matching:

- **BERT similarity (0.15)** and **DBAName_formation (0.15)** receive the highest weights because:
    - BERT captures semantic relationships, connecting “Apple” to “iPhone maker” even with no character overlap
    - DBAName_formation recognizes acronym patterns (e.g., “B of A” → “Bank of America”)
- **Middle-tier algorithms (0.10)** like Jaro-Winkler and token_sort_ratio handle common variations like minor misspellings and word order changes
- **Specialized algorithms (0.05)** like soundex and fuzzy_levenshtein are useful in specific scenarios but might produce false positives in general matching

### What Would Happen If We Changed These Values?

- **Increasing BERT’s weight** (e.g., to 0.30): Would prioritize semantic understanding but might miss character-level variations or typographical errors
- **Decreasing BERT’s weight** (e.g., to 0.05): Would underutilize semantic understanding, missing connections between abbreviations and full names
- **Removing weights entirely**: Would treat all algorithms equally, ignoring their different strengths and weaknesses for merchant matching

The base weights create a balanced foundation that prevents any single algorithm from dominating the matching process.

## 3. Category-Specific Weight Adjustments

```python
if primary_category == 'Restaurant':
    weights['bert_similarity'] = 0.25
    weights['fuzzy_levenshtein'] = 0.15
elif primary_category == 'Banking':
    weights['DBAName_formation'] = 0.25
    weights['enhanced_DBAName_formation'] = 0.25
    weights['bert_similarity'] = 0.20
elif primary_category == 'Government':
    weights['bert_similarity'] = 0.25
    weights['DBAName_formation'] = 0.20
    weights['token_sort_ratio'] = 0.15
# Additional categories follow similar patterns
```

### Why Adjust Weights by Category?

Different industries follow distinct naming conventions:

- **Restaurants** often use creative spellings (e.g., "Krispy Kreme," "Chick-fil-A"), making BERT's semantic understanding and fuzzy matching more valuable
- **Banking institutions** commonly use formal abbreviations (e.g., "BofA" for "Bank of America"), making DBAName_formation crucial
- **Government entities** use standardized acronyms (e.g., "DOJ" for "Department of Justice"), necessitating both semantic understanding and acronym recognition

### What If We Removed Category Adjustments?

Without category-specific adjustments, we would see: - More false positives for restaurants due to inadequate handling of creative spellings - Lower matching accuracy for banks and government entities due to insufficient emphasis on acronym formation - Overall decreased matching performance across diverse merchant types

## 4. Length-Based Weight Adjustments

```
# For very short DBANames (2-3 chars)
if 2 <= DBAName_len <= 3:
    weights['DBAName_formation'] = 0.25
    weights['enhanced_DBAName_formation'] = 0.20
    weights['bert_similarity'] = 0.15
    weights['contains_ratio'] = 0.15

# For longer DBANames
elif DBAName_len >= 4:
    weights['bert_similarity'] = 0.25
    weights['token_sort_ratio'] = 0.15

# For very long full names
if RawTransactionName_len > 30:
    weights['bert_similarity'] = 0.30
    weights['tfidf_cosine'] = 0.15
```

### Why Adjust Weights by Length?

The length of merchant names provides valuable clues about their structure:

- **Very short names (2-3 characters)** are typically acronyms (e.g., “BOA”), so DBAName_formation and enhanced_DBAName_formation receive higher weights
- **Longer names (≥4 characters)** contain more semantic information, making BERT and token_sort_ratio more valuable
- **Very long full names (>30 characters)** often include additional descriptive information, making semantic understanding (BERT) and keyword matching (TF-IDF) especially important

### What If We Removed Length Adjustments?

Without length-based adjustments: - Very short acronyms would be inadequately matched to their full names - Longer names would receive insufficient semantic analysis - The system would treat all merchant names the same regardless of their length characteristics

## 6. Abbreviation Dictionaries

```python
def _get_abbreviation_dictionary(self):
    """Get comprehensive abbreviation dictionary"""
    return {
        # Banking & Financial Institutions
        'bofa': 'bank of america', 'b of a': 'bank of america',
        'boa': 'bank of america', 'bac': 'bank of america',
        'jpm': 'jpmorgan chase', 'jpm chase': 'jpmorgan chase',
        # Many more abbreviations follow
    }
```

### Why Hard-Code Abbreviations?

Business abbreviations are often: - **Non-intuitive**: “AMEX” for “American Express” can’t be derived from simple letter extraction - **Ambiguous**: “BofA” could be “Bank of America” or “Banana of America” without domain knowledge - **Industry-standard**: These are established conventions that don’t change frequently

During preprocessing, the system expands abbreviations according to these dictionaries, allowing for better matching between abbreviated and full names.

### What If We Removed Abbreviation Dictionaries?

Without abbreviation dictionaries: - Common abbreviations would fail to match their expanded forms - The system would rely entirely on BERT to make these connections, which may not always succeed - Industry-specific abbreviations would be particularly challenging to match

## 6. Score Caps and Thresholds

```
# Apply category-based constraints according to business rule
if categories_match:
    # If categories match exactly and name similarity is good
        (score > 0.60),
    # ensure score is at least 0.75
    if boosted_score > 0.60:
        final_score = max(0.75, boosted_score)
    else:
        # Categories match but names are too dissimilar, keep
        original score
        final_score = boosted_score
else:

        # If categories don't match, cap the score at 0.75
        regardless of name similarity
    final_score = min(0.75, boosted_score)
```

### Why Apply Score Caps and Thresholds?

These rules enforce business logic about match confidence:

- **Cross-category cap (0.75)**: Prevents high confidence in matches between different types of businesses
- **Same-category minimum (0.75)**: Ensures reasonable confidence when category matches and names are similar
- **Pattern boost cap (1.6)**: Prevents pattern recognition alone from creating overconfidence

### What If We Removed Score Caps?

Without score caps and thresholds: - The system might report high confidence for cross-category matches, leading to false positives - Similar names in the same category might receive unnecessarily low scores - Pattern recognition might dominate other signals, causing imbalanced matching

## 7. Conclusion and Future Directions

Our merchant matching system combines the semantic power of BERT MPNet with traditional string matching algorithms, enhanced by domain knowledge encoded through hard-coded weights and rules. This approach provides several advantages:

1. **Efficiency**: By using hard-coded weights instead of training a fully dynamic model, we avoid computational overhead
2. **Domain expertise integration**: Years of experience with merchant name patterns are directly incorporated
3. **Adaptability**: The system adjusts its strategy based on merchant category and name length
4. **Robustness**: Multiple algorithms work together, preventing any single approach from dominating

## Future Enhancements

As we gather more labeled training data, we could consider: 1. **Learning optimal weights**: Train a model to dynamically determine the most effective weights 2. **Adding more specialized algorithms**: Develop matching techniques for specific industries 3. **Enhanced pattern recognition**: Identify additional patterns that indicate matching merchants 4. **Feedback loop**: Incorporate corrections from users to continuously improve matching accuracy

The current hard-coded approach provides a strong foundation that balances efficiency with accuracy, leveraging both modern NLP techniques and domain-specific knowledge to solve the challenging problem of merchant name matching.

# Proof of Concept for Hard-Coded Merchant Matching System

## 1. Executive Summary

This report provides a proof of concept and justification for the current hard-coded approach employed in the merchant matching system. The decision to utilize hard-coded elements, rather than a fully dynamic system, stems from several critical factors, including the lack of extensive labeled training data, computational resource constraints, and the strategic advantage of directly incorporating years of valuable domain expertise in text matching. While the system leverages the semantic understanding capabilities of the BERT MPNet model, it does so within a broader framework enriched by carefully calibrated, knowledge-driven components. This hybrid approach has proven effective in achieving robust and accurate merchant matching without the need for extensive machine learning training on a large dataset. The hard-coded elements are not arbitrary; they are the result of deep analytical understanding of merchant naming conventions and the performance characteristics of various matching algorithms.

## 2. Introduction: The Challenge of Merchant Matching

Accurate merchant matching is a fundamental requirement for numerous business operations, including comprehensive spend analysis, efficient vendor management, and the proactive detection of fraudulent activities.[1] The challenge lies in the inherent inconsistency of merchant names as they appear in diverse datasets. These names often contain abbreviations, misspellings, variations in formatting, and the use of nicknames or short forms, making reliance on exact matching techniques inadequate.[1] To overcome these complexities, sophisticated techniques beyond simple string comparison are necessary. Fuzzy matching, which considers degrees of similarity between strings, emerges as a crucial approach in this context.[4] Traditional rule-based or exact matching algorithms often fail to capture the nuances present in real-world merchant names, highlighting the need for more adaptable and intelligent systems.[3] The current merchant matching system addresses this need by employing a strategic combination of hard-coded elements and the BERT MPNet model, a powerful tool for understanding the meaning of text. This report aims to provide a detailed explanation and robust justification for the hard-coded components that form a vital part of this system.

## 3. Why Hard-Coding? Addressing Current Limitations

The decision to incorporate hard-coded values into the merchant matching system was a deliberate choice made to address specific limitations associated with a purely dynamic, data-driven approach within the current operational context.

The training of a fully dynamic model hinges on the availability of a large, high-quality dataset comprising correctly matched merchant name pairs.[7] Such a dataset, which would enable a machine learning model to learn the intricate patterns and rules governing merchant name variations, is currently either unavailable or insufficient in size and quality to train the entire matching logic effectively. Without a substantial volume of accurately labeled data, a dynamic model would likely exhibit suboptimal performance, leading to inaccurate matches and a failure to identify valid relationships between merchant names.[9] Machine learning models learn by identifying patterns in the data they are trained on. If the training data lacks comprehensive examples of the variations and nuances inherent in merchant names, the model will struggle to generalize effectively to unseen data.

Furthermore, a purely BERT-driven approach, where the full power of a large Natural Language Processing (NLP) model like BERT MPNet is dynamically unpacked and utilized for every single merchant name comparison, can be computationally prohibitive.[10] This process demands significant Central Processing Unit (CPU) resources, potentially leading to performance bottlenecks and increased infrastructure costs. While BERT MPNet excels at understanding the semantic meaning of text, applying it universally across all comparisons might not be the most efficient strategy, especially for cases where simpler string-based comparisons could suffice. The computational intensity of processing every comparison through a large language model could strain the existing infrastructure and impact the overall speed and responsiveness of the merchant matching system.[7]

In situations where extensive labeled data is lacking, and computational resources are a concern, the strategic incorporation of domain expertise becomes paramount.[9] The hard-coded elements within the merchant matching system are precisely designed to embody years of accumulated experience and a deep understanding of merchant naming conventions and the most effective matching strategies. Hard-coding allows for the direct embedding of this invaluable domain knowledge into the system's logic, enabling it to perform robustly and accurately even in the absence of a large, labeled training dataset.[16] This approach leverages the insights gained over time regarding which matching algorithms and rules are most effective for different types of merchant names and their variations.

In conclusion, the strategic use of hard-coding provides a stable, predictable, and computationally efficient solution tailored to the current circumstances. It allows the merchant matching system to perform effectively by directly leveraging the expertise of domain specialists, ensuring a level of accuracy and reliability that might be challenging to achieve with a purely dynamic approach given the existing constraints.[3]

## 4. Deconstructing the Hard-Coded Elements: Logic and Rationale

The merchant matching system strategically employs several categories of hard-coded elements, each designed with a specific purpose and informed by deep domain understanding.

### 4.1 Base Weights in Similarity Algorithms:

The system utilizes a collection of similarity algorithms, each assigned a base weight reflecting its general importance in merchant name comparison. These algorithms and their base weights are: jaro_winkler (0.10), damerau_levenshtein (0.05), tfidf_cosine (0.05), jaccard_bigram (0.05), soundex (0.05), token_sort_ratio (0.10), contains_ratio (0.10), fuzzy_levenshtein (0.05), trie_approximate (0.10), bert_similarity (0.15), aho_corasick (0.05), and DBAName_formation (0.15). Each of these algorithms serves a specific purpose in evaluating the similarity between two merchant names.[1] For instance, Jaro-Winkler is effective at identifying phonetic similarity, while TF-IDF cosine measures the similarity based on the frequency of terms. The specific weight assigned to each algorithm is not arbitrary but is based on years of experience in determining their general reliability and suitability for a broad range of merchant name comparisons.

Notably, BERT similarity (0.15) and DBAName_formation (0.15) are assigned the highest base weights. BERT's semantic understanding is crucial for accurately matching abbreviations and capturing the underlying meaning of business names, going beyond mere literal string similarity.[20] For example, BERT can recognize the relationship between "NAB" and "National Australia Bank" even with limited character overlap. This capability is vital in the context of merchant names, where abbreviations are frequently used. DBAName_formation is also highly weighted due to its ability to recognize acronym formation patterns, which are prevalent in business names.

A middle tier of weights (0.10) is assigned to Jaro-Winkler, token_sort_ratio, contains_ratio, and trie_approximate. These algorithms have proven to be consistently reliable for comparing business names that exhibit minor variations, such as slight misspellings or differences in word order.[1]

The remaining algorithms (damerau_levenshtein, tfidf_cosine, jaccard_bigram, soundex, fuzzy_levenshtein, and aho_corasick) are assigned lower weights of 0.05 each. While these algorithms are valuable in specific scenarios, they tend to be more specialized or have a higher likelihood of producing false positives in general merchant name matching.[1]

These base weights collectively represent a balanced strategy, ensuring that no single matching technique unduly influences the overall similarity score. This distribution reflects expert knowledge about the relative performance of each algorithm in the context of merchant name matching, particularly in the absence of extensive training data. The weighted combination harnesses the strengths of different algorithms, leading to a more robust and accurate matching system capable of handling a diverse range of name variations.

### 4.2 Category-Specific Weight Adjustments:

The merchant matching system incorporates category-specific weight adjustments to account for the unique naming patterns observed in different industries. For instance, if the primary category of a merchant is identified as 'Restaurant', the weight for bert_similarity is increased to 0.25, and the weight for fuzzy_levenshtein is increased to 0.15. This adjustment recognizes that restaurant names often employ creative spellings and non-standard phrasing to enhance memorability and branding.[1] Algorithms that can effectively handle phonetic similarity and minor spelling variations are therefore more critical for accurately matching restaurant names.

Similarly, for merchants categorized as 'Banking' or 'Government', the weight for DBAName_formation is boosted to 0.25. This reflects the common practice in these sectors of using formal abbreviations and standardized acronyms in their names. An algorithm specifically designed to identify these acronyms and abbreviations is thus more likely to yield accurate matches for banking and government entities.

These category-specific adjustments enable the system to tailor its matching approach based on the industry of the merchant, without requiring the development and maintenance of separate models for each industry. This targeted adjustment of weights allows for a more nuanced and accurate matching process that takes into account the distinct naming conventions prevalent in different business domains.

### 4.3 Length-Based Weight Adjustments:

The length of a merchant name can also provide valuable clues about its structure and the most effective matching techniques. The system incorporates length-based weight adjustments to capitalize on this observation. For very short merchant names, specifically those with a DBAName_len between 2 and 3 characters, the weights for DBAName_formation (0.25), enhanced_DBAName_formation (0.20), bert_similarity (0.15), and contains_ratio (0.15) are increased. The rationale behind this is that very short names are often acronyms, making the DBAName_formation algorithms particularly relevant. Additionally, the contains_ratio algorithm can effectively identify matches where one short name is contained within another.

Conversely, for longer merchant names, the weight for bert_similarity is increased to a range of 0.25 to 0.30. This adjustment acknowledges that longer names typically contain more semantic information, making BERT's understanding of meaning more valuable for accurate matching. In these cases, the overall semantic context becomes more important than just character-level similarity.

This length-based adjustment demonstrates an understanding that different matching techniques perform optimally for names of varying lengths. A purely dynamic system would require extensive training to learn these relationships implicitly, whereas the hard-coded adjustments directly apply this domain knowledge, leading to more immediate and effective matching.

**4.4 Pattern Boost Factors:**

The merchant matching system includes pattern boost factors to specifically handle common naming patterns that are strong indicators of a match. These patterns include inverted agency structures (e.g., "Department of Treasury" vs. "Treasury Department"), bank name inversions (e.g., "Bank of America" vs. "America Bank"), and ampersand abbreviations (e.g., "Johnson & Johnson" becoming "J&J"). For each identified pattern, a specific boost factor is added to the similarity score. For instance, an inverted agency structure triggers a 35% boost, as does a bank name inversion. An ampersand abbreviation results in a 30% boost.

These specific boost values (0.35, 0.30, etc.) are carefully calibrated to increase the confidence in a match when these patterns are observed, without overwhelming other signals from the similarity algorithms. They represent the empirically determined reliability of these specific naming patterns as indicators of a true match, based on extensive observation of merchant naming conventions.

**4.5 Abbreviation Dictionaries:**

The system utilizes a comprehensive, hard-coded dictionary of common business abbreviations to provide direct mappings for known short forms. This dictionary includes entries such as {'bofa': 'bank of america', 'b of a': 'bank of america', 'boa':

'bank of america', 'bac': 'bank of america'}. This hard-coding is essential for several reasons. Many of these abbreviations are established industry standards, and their recognition is crucial for accurate merchant matching. Furthermore, a significant number of business abbreviations are non-intuitive and would not be easily deciphered by general language models without specific training. For example, "AMEX" for "American Express" is a common but non-obvious abbreviation. Additionally, some abbreviations can be ambiguous, and their correct resolution requires domain-specific knowledge. While BERT MPNet might be able to infer some of these relationships given enough context, the explicit mapping provided by the hard-coded dictionary ensures a higher degree of accuracy for these critical cases. The dictionary effectively encodes human knowledge about common business abbreviations, knowledge that would necessitate a vast amount of training data for a purely dynamic system to learn comprehensively.

### 4.6 Score Caps and Thresholds:

The merchant matching system employs score caps and thresholds to implement business rules regarding the level of confidence in a match. These rules are designed to prevent false positives and ensure that the final match score aligns with business expectations. For instance, if two merchant names belong to different primary categories, the final boosted score is capped at 0.75, regardless of how high the similarity score might otherwise be. This rule prevents the system from expressing high confidence in matches across fundamentally different types of businesses. Conversely, if two merchant names fall within the same category and their boosted similarity score is above 0.60, the final score is set to a minimum of 0.75. This ensures a reasonable level of confidence for matches within the same category that exhibit a certain degree of string similarity. Additionally, the pattern boost applied in the system is capped at 1.6 (a 60% increase) to prevent pattern matching alone from leading to an overly high confidence score that might not be supported by other evidence. These specific values (0.60, 0.75, 1.6) represent confidence thresholds that have been determined through rigorous testing and informed by domain expertise to establish reasonable boundaries for match confidence and minimize the occurrence of false positives.

### 4.7 Hard-Coded Preprocessing Rules:

The system incorporates hard-coded preprocessing rules to address specific, recurring patterns in merchant names that general algorithms might overlook. One example is the handling of the "Mc" prefix commonly found in names like McDonald's. The rule checks if any word in the merchant name contains "mc" (case-insensitive) and, if so, transforms variations like "mcd", "mcds", or "mcdon" into the standard "mcdonalds". This specific handling addresses the unique convention of the "Mc" prefix. Another preprocessing rule involves the removal of common business suffixes such as "Inc" or "LLC". These suffixes often do not contribute significantly to the core identity of the merchant and can introduce noise in the matching process.[2] Finally, the system includes a rule for apostrophe normalization, converting possessive forms to a standard representation to ensure consistent comparison. These preprocessing rules encode domain-specific knowledge about common variations and irrelevant elements in merchant names, knowledge that would require a substantial amount of training data for a dynamic system to learn and apply effectively.

## 5. The Role of BERT MPNet: A Complementary Approach

Within the merchant matching system, BERT MPNet plays a crucial role in providing semantic understanding.[20] Unlike traditional string matching algorithms that primarily focus on character-level or phonetic similarity, BERT captures the underlying meaning of words and phrases. This capability allows the system to recognize that "National Australia Bank" and "NAB" are related, even though they share limited character overlap. BERT achieves this by having been pre-trained on a massive corpus of text, enabling it to learn contextualized word embeddings that represent the semantic relationships between words.

However, BERT is implemented as a complement to, rather than a replacement for, traditional string matching methods. While BERT excels at identifying semantic similarity, it may not be as effective at capturing minor typographical errors or subtle phonetic variations that traditional algorithms are designed to handle. Therefore, the hard-coded weights within the system are carefully calibrated to balance the semantic power of BERT with the reliability of traditional algorithms for character-level matching.

The default weight assigned to BERT similarity (0.15) provides it with a significant influence on the final match score without allowing it to dominate the results. Furthermore, the system dynamically adjusts the weight of BERT similarity for longer merchant names, increasing it to a range of 0.25 to 0.30. This adjustment recognizes that for longer names, which typically contain more semantic information, BERT's understanding of meaning becomes increasingly important for accurate matching.

From a computational perspective, utilizing a multi-algorithm approach with strategically applied hard-coded weights offers a more efficient solution compared to running every merchant name comparison through the computationally intensive

BERT model.[10] This hybrid approach allows the system to leverage the strengths of BERT for cases where semantic understanding is critical while relying on more efficient algorithms for other types of comparisons.

## 6. Why These Specific Weights Empower Other Values

The base weights assigned to the various similarity algorithms are not arbitrary; they are the result of careful calibration based on extensive domain expertise in merchant name matching.[9] This calibration reflects years of experience in understanding the relative importance and reliability of each algorithm for this specific task. Through empirical observation, the team has identified which algorithms consistently yield accurate matches and which are more susceptible to errors or false positives. The assigned weights directly reflect this accumulated knowledge.

The higher weights allocated to BERT similarity and DBAName_formation are crucial in empowering the system to effectively address key challenges in merchant name matching, such as recognizing semantic similarity and identifying abbreviations. These are aspects where traditional string-based algorithms often fall short. By giving these algorithms more weight, the system prioritizes these vital aspects of comparison.

Furthermore, the category-specific and length-based adjustments to the weights serve to refine the impact of the base weights, allowing the system to adapt its matching strategy based on the specific characteristics of the merchant names being compared. These adjustments ensure that the most relevant algorithms are given greater importance in different contexts, leading to more accurate and nuanced matching results.

The design of the weighting system also aims to prevent any single algorithm from exerting a disproportionate influence on the final match score. This balanced approach ensures that the system considers multiple facets of similarity, leading to a more robust and reliable overall matching process. The specific weights chosen represent an optimal balance identified through expert knowledge and testing, maximizing the accuracy and effectiveness of the merchant matching system.

## 7. Understanding Weights in Pre-trained Models (BERT MPNet)

In a pre-trained language model like BERT MPNet, "weights" refer to the parameters that the model learns during its initial training phase on a massive dataset of text.[21] These weights are numerical values that represent the model's understanding of language patterns, the relationships between words, and the underlying semantic meaning.[20] During the pre-training process, BERT MPNet is exposed to billions of words and learns to perform tasks like predicting masked words and understanding the relationships between sentences. The weights are the outcome of this learning process, capturing the model's internalized knowledge of language.

It is important to note that these weights are not typically tied to specific individual words (tokens) but rather to word types and the contextual relationships in which they appear.[25] This means that the same word can have different representations within the model depending on the other words in the sentence. BERT's architecture, which utilizes a mechanism called "attention," allows it to understand the context of each word and adjust its representation accordingly.

This understanding of weights in BERT MPNet is directly relevant to why the bert_similarity algorithm in the hard-coded system is assigned a specific weight (initially 0.15, with adjustments based on category and length). This weight determines the extent to which BERT's learned semantic understanding influences the final match score. A weight of 0.15 signifies that the system considers the semantic similarity identified by BERT as a significant factor in determining whether two merchant names are a match. This specific value has been determined through experience to provide a good balance with other factors captured by the traditional algorithms.

The table below summarizes the performance and characteristics of several Sentence Transformer models, including 'all-mpnet-base-v2', which is a likely candidate for the BERT MPNet model used in the system given its strong performance in sentence embedding and semantic search tasks.[10]

| Model Name | Performance Sentence Embeddings (14 Datasets) | Performance Semantic Search (6 Datasets) | Avg. Performance...source |

This table demonstrates the rationale behind potentially selecting 'all-mpnet-base-v2'. Its high average performance across various NLP tasks, particularly in sentence embeddings and semantic search, makes it well-suited for the merchant matching system's needs. The balance between performance, speed, and model size is also a crucial consideration, and 'all-mpnet-base-v2' strikes a favorable compromise among these factors.

## 8. Conclusion

The strategic utilization of hard-coded elements in the merchant matching system is a well-founded decision driven by the need to leverage years of invaluable domain expertise in text matching [9], address the current limitations of extensive labeled training data, and provide a computationally efficient solution compared to a purely dynamic approach relying solely on large language models.[10] The hard-coded weights and rules are not arbitrary; they are meticulously calibrated to create a balanced integration of multiple matching approaches, account for specific industry patterns and name lengths, and enforce essential business rules regarding the confidence in a match. The BERT MPNet model plays a critical complementary role by providing semantic understanding, enhancing the system's ability to recognize non-literal matches. The current hard-coded approach, therefore, offers a robust and effective solution for merchant matching within the existing constraints. As the availability of labeled training data improves and computational resources evolve, future explorations into dynamic weight learning could be considered to further enhance the system's capabilities.

## Works cited

1. Name Matching Techniques: Useful Algorithms, Their Problems, & Absolute Solutions, accessed April 16, 2025, https://singlequote.blog/name-matching-techniques-useful-algorithms-their-problems-absolute-solutions/
2. What Is Merchant Normalization? | Machine Learning - Coupa, accessed April 16, 2025, https://www.coupa.com/blog/what-merchant-normalization/
3. Name Normalization: Matching Companies, Vendors, Suppliers - RecordLinker, accessed April 16, 2025, https://recordlinker.com/name-normalization-matching/
4. A Business Guide to Fuzzy Matching - Firstlogic, accessed April 16, 2025, https://firstlogic.com/insights/a-business-guide-to-fuzzy-matching
5. Try Our Name Matching Algorithm API - Width.ai, accessed April 16, 2025, https://www.width.ai/name-matching-algorithm-api
6. Name Matching Software vs Algorithms: Which is Best for Your Business? - Data Ladder, accessed April 16, 2025, https://dataladder.com/name-matching-software-vs-algorithms/
7. Methods of Name Matching - Fuzzy Matching Techniques | Babel Street, accessed April 16, 2025, https://www.babelstreet.com/blog/fuzzy-name-matching-techniques
8. Top 6 Name Matching Algorithm, How To Scale Your Solution & Tutorial In Python, accessed April 16, 2025, https://spotintelligence.com/2023/07/10/name-matching-algorithm/
9. The Role of Domain Knowledge in Machine Learning: Why Subject Matter Experts Matter, accessed April 16, 2025, https://machinelearningmastery.com/role-domain-knowledge-machine-learning/
10. Pretrained Models — Sentence Transformers documentation, accessed April 16, 2025, https://www.sbert.net/docs/sentence_transformer/pretrained_models.html
11. machinelearningmastery.com, accessed April 16, 2025, https://machinelearningmastery.com/role-domain-knowledge-machine-learning/#:~:text=How%20does%20domain%20knowledge%20enhance,effective%20in%

20their%20intended%20applications.
12. AI in Context: Harnessing Domain Knowledge for Smarter Machine Learning - MDPI, accessed April 16, 2025, https://www.mdpi.com/2076-3417/14/24/11612
13. Domain Expertise vs Machine Learning Debate - Experfy Insights, accessed April 16, 2025, https://resources.experfy.com/ai-ml/domain-expertise-vs-machine-learning-debate/
14. 8 Domain Knowledge - Supervised Machine Learning for Science, accessed April 16, 2025, https://ml-science-book.com/domain.html
15. Domain Expert is an Essential Block of a Robust ML System | Towards Data Science, accessed April 16, 2025, https://towardsdatascience.com/domain-expert-is-an-essential-block-of-a-robust-ml-system-a29fd1576832/
16. Is Domain Knowledge Important for Machine Learning? - KDnuggets, accessed April 16, 2025, https://www.kdnuggets.com/2022/07/domain-knowledge-important-machine-learning.html
17. Role of AI Domain Expertise in Choosing the Right Solution - IT Convergence, accessed April 16, 2025, https://www.itconvergence.com/blog/the-role-of-domain-expertise-in-choosing-ai-ml-solutions/
18. 1 – The Importance of Domain Knowledge – Machine Learning Blog | ML@CMU, accessed April 16, 2025, https://blog.ml.cmu.edu/2020/08/31/1-domain-knowledge/
19. Applying Domain Expertise and Machine Learning Techniques to Provide Actionable Solutions - MPR Associates, accessed April 16, 2025, https://www.mpr.com/case-studies/applying-domain-expertise-and-machine-learning-techniques-to-provide-actionable-solutions/
20. Sentence Transformers: Meanings in Disguise - Pinecone, accessed April 16, 2025, https://www.pinecone.io/learn/series/nlp/sentence-embeddings/
21. CrisisTransformers: Pre-trained language models and sentence encoders for crisis-related social media texts - arXiv, accessed April 16, 2025, https://arxiv.org/pdf/2309.05494
22. sentence-transformers/all-mpnet-base-v2 - Hugging Face, accessed April 16, 2025, https://huggingface.co/sentence-transformers/all-mpnet-base-v2
23. MPNet - Hugging Face, accessed April 16, 2025, https://huggingface.co/docs/transformers/en/model_doc/mpnet
24. RUCAIBox/PLMPapers: A paper list of pre-trained language models (PLMs). - GitHub, accessed April 16, 2025, https://github.com/RUCAIBox/PLMPapers
25. How pre-trained weights in the BERT can help the fine tuning task? - Cross Validated, accessed April 16, 2025, https://stats.stackexchange.com/questions/517802/how-pre-trained-weights-in-the-bert-can-help-the-fine-tuning-task
26. Assigning weights during testing the bert model - Stack Overflow, accessed April 16, 2025,

https://stackoverflow.com/questions/65925640/assigning-weights-during-testing-the-bert-model