Generation Of Contextual Vector Embeddings For Indian Address And Clustering

Awadhoot Khutwad
SCTR's Pune Institute of Computer Technology
awadhootk6@gmail.com

Abstract—**In this paper, we introduce a novel method for parsing e-commerce customer addresses in developing nations like India, where address formats vary widely. Drawing inspiration from recent advancements in Natural Language Processing (NLP), we propose a multi-step approach involving edit distance and phonetic algorithms for address preprocessing. We then explore different techniques, including Word2Vec with TF-IDF, BiLSTM, and BERT, to create vector representations of addresses. Once pre-trained, the RoBERTa model can be fine-tuned for various downstream tasks in supply chain management, such as pin code suggestion and geo-coding, even with limited labeled data. This research represents an effort in exploring natural language algorithms to understand customer addresses in the e-commerce domain.**

Index Terms—**Address pre-processing, Word2Vec, TF-IDF, BiLSTM, RoBERTa, Soundex, Levenshtein, Cosine Similarity.**

I. Introduction

Customers in a developing nations like India tend to follow no fixed format while entering addresses. Parsing such addresses is challenging because of a lack of inherent structure or hierarchy. It is imperative to understand the language of addresses, so that customers with similar/nearby addresses can be grouped together. Overall this capability will help us to design better machine learning algorithms. The problem statement explores how recent advances in Natural Language Processing (NLP) can be used towards understanding custom Indian addresses. The solution and approach need to address various pre-processing and standardization methods using various NLP techniques. The requirement is to explore the following aspects.

1) Identify a corpus of Urban and Rural Indian addresses
2) Devise methods to Pre-process and standardize Urban and Rural addresses separately
3) Use the pre-processed addresses to generate contextual vector embeddings using Deep Learning and Neural Networks
4) Use the generated vector embeddings for clustering similar/nearby addresses

In developing nations like India, the lack of standardized address formats presents a formidable challenge for machine processing of manually entered addresses, particularly in the realm of e-commerce. Customers often input shipping addresses with varying structures and idiosyncratic interpretations of correctness, complicating the task of routing shipments for last-mile delivery. Consider the diverse examples of addresses entered by customers:

1) *'XXX , AECS Layout, Geddalahalli, Sanjaynagar main Road Opp. Indian Oil petrol pump, Ramkrishna Layout, Bengaluru Karnataka 560037'*
2) *'XXX, B-Block, New Chandra CHS, Veera Desai Rd, Azad Nagar 2, Jeevan Nagar, Azad Nagar, Andheri West, Mumbai,*

These addresses highlight several challenges unique to developing countries:

- Lack of Prescribed Structure: Unlike addresses in more developed regions, those in developing countries lack a standardized structure, leading to inconsistencies and variations.
- Geographical Ambiguity: Not all addresses are readily associated with a specific geographical location, complicating the task of mapping addresses to physical locations.
- Ethnic Diversity: With multiple ethnic groups within countries like India, address elements such as area names and house structures vary significantly across regions.
- Variable Length and Complexity: Addresses in developing countries can range from 2 to 20 words, often including additional information such as directions, landmarks, and contact details, further increasing complexity.
- Challenges with Postal Codes: Postal codes, known as PIN codes in India, pose additional challenges due to limited literacy levels and rapid locality growth, leading to ambiguity regarding geographical zones.

These complexities pose significant challenges for machine learning-based solutions, including address classification, clustering, and fraud detection. E-commerce companies, in particular, face various forms of fraud, including reseller fraud and fraudulent claims related to missing items or mis-shipments. Resellers exploit online discounts/offers by purchasing items and selling them offline for profit, often altering address patterns to circumvent purchase limits. Machine learning models play a crucial role in identifying fraudulent transactions, leveraging signals such as variations in customer addresses with fraudulent intent.

II. Related Work

[1] Propose BERT architecture, which stands for Bidirectional En- coder Representations from Transformers. They demonstrate that BERT obtains state-of-the-art performance on several NLP tasks like question answering, classification etc. Liu et al.

[2] Discuss and propose techniques to cluster Indian addresses using Word2Vec, TF-IDF, RoBERTa and Bi-LSTM approaches.

[5] Show methods to classify textual address into geographical sub-regions for shipment delivery using Machine Learning. Their work mainly involves address pre-processing, clustering and classi- fication using ensemble of classifiers to classify the addresses into sub-regions.

[6] Propose methods to pre-process addresses and learn their latent representations using different approach - sequential network (LSTM).

[7] Discuss the challenges with address data in Indian context and propose methods for efficient large scale address clustering using conventional as well as deep learning techniques. They ex- periment with different variants of Leader clustering using edit distance, word embedding etc. For detecting fraud addresses over e- commerce platforms and reduce operational cost, Babu and Kakkar.

III. Methodology

The problem-solving involved three main phases of work:

I. Address Pre-processing, finding and replacing with leader token.

II. Generation of Contextual Vector Embeddings.

III. Comparison and Analysis of the various methods and Clustering.

A. *Address Pre-processing*

1) Tokenization: Tokenizing involves dividing the sentence into single-word tokens to facilitate checking for spell variants. This step is crucial for further processing.
For example:
Before Tokenizing: [Tulip apartment Laxmi nagar]
After Tokenizing: [[Tulip], [apartment], [Laxmi], [nagar]]

2) Custom Token Removal: Special characters like whitespaces, numbers, words with lengths greater than 20 or less than 4, and lowercase stopwords are removed. This process helps in standardizing the address format and removing irrelevant information.

- Lowercasing to ensure uniformity.
- Removal of any character that is not an alphabet.
- Words with lengths less than 4 or greater than 20 are excluded.
- Special characters are removed.

For example:
Before Preprocessing: [#Park Street 45]
After Preprocessing: [parkstreet]

3) Lemmatization: Lemmatization groups together different inflected forms of words to consider them as a single item.
For example:
Before Preprocessing: [Walk , Walks, Walking, Walked]
After Preprocessing: [Walk]

4) Clustering of similar words:

- *Soundex Algorithm*
  Soundex is amongst the early algorithms designed for phonetics-based matching which is still used in US Census. It generates a 5 digit code for a string based on its phonetic sound.
  For example:
  'primary': P6560, 'primery': P6560, 'meethapur': M3160

- *Levenshtein Distance Algorithm*
  Levenshtein distance between two strings is defined as the minimum number of characters needed to insert, delete or replace in a given string string1 to transform it to another string string2.
  For example:
  Levenshtein_ratio('Pune', 'Poona') = 0.44

- *Custom Algorithm - Soundex + Levenshtein*
  Advantages of the Custom Algorithm:
  1) **Combination of Phonetic and String Similarity:** This algorithm combines two types of similarity measures: phonetic similarity (Soundex) and string similarity (Levenshtein). This combination is more robust in capturing variations in pronunciation and spelling.
  2) **Fine-Tuning with Levenshtein Distance:** By applying a Levenshtein distance threshold after Soundex grouping, this algorithm can filter out only those words that are phonetically similar and have a high degree of string similarity. This ensures a more precise selection of similar words.
  3) **Efficiency:** Although the first algorithm involves two steps (Soundex and Levenshtein), it potentially reduces the number of comparisons by grouping similar words first using Soundex. This grouping can significantly reduce the computational burden when compared to pairwise comparisons of all words.
  4) **Customization:** This algorithm allows for customization of the Levenshtein threshold, providing flexibility in defining the level of similarity required between words.

  For example:
  Let the corpus be: ['pune', 'puna', 'poone', 'poona' ,'pony']
  Group: ['pune', 'puna', 'poone']

- *Replace with Leader Token*
  Leader token is simply calculated as the word with maximum frequency in a group of similarly identified words. Words are replaced with leader tokens using regex functions provided by PySpark.

B. *Generation of Contextual Vector Embeddings*

Vector embeddings are a way to convert words and sentences and other data into numbers that capture their meaning and relationships. They represent different data types as points

Original Sentence	**Replaced Token**	**Token replaced**
Ganpat Patil Nagar Galli No 14 SOHARADANGI KUMARIPUR KATIHAR Pune Hadapsar NEAR AT PRIMARY SCHOOL	ganpati patil nagar gali no 14 soharadangi kumari- pur kati- har Pune hadap- sar near at primary school	galli is replaced by gali
Pune HADAPASAR PUNE sadar	pune hadap- sar Pune sadar	HADAPASAR is replaced by hadap- sar

TABLE I: Results after Preprocessing

in a multidimensional space, where similar data points are clustered closer together.

I. Word2Vec + TF-IDF (and variations)
II. Bi-LSTM Neural Network
III. RoBERTa Model

1) Word2Vec + TF-IDF:

- *Word2Vec:* The main objective of using Word2Vec is to capture and represent the semantic meaning of words in a vector space, allowing for more efficient and meaningful text analysis. This technique helps with tasks like text classification, information retrieval, sentiment analysis, and recommendation systems by enabling the understanding of word relationships and context in natural language text. Word2Vec is a popular word embedding technique implemented in the Gensim library. It learns vector representations of words from large text corpora, capturing semantic relationships between words. These vectors can be used for various natural language processing tasks, like similarity calculations or input for machine learning models. Word2Vec has 2 methods of generating feature vectors:
  1) **Continuous Bag-of-Words (CBOW)**
     Continuous Bag of Words (CBOW) is a Word2Vec model architecture that aims to predict a target word based on its context. In CBOW, the model takes a fixed-size context window of surrounding words as input and predicts the target word in the center of the window. The model is trained by maximizing the probability of predicting the target word given its context words. CBOW is known for its efficiency in training and is particularly effective for frequent words.
  2) **Skip-gram**:
     Skip-gram is another Word2Vec model architecture that works in the opposite way to CBOW. Instead of predicting a target word based on its context, Skip-gram predicts the context words given a target word. The model learns to generate context words that are likely to appear together with the target word. Skip-gram is especially useful for capturing semantic relationships between words and performs well with less frequent words.

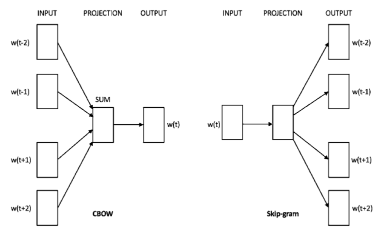


Fig. 1: CBOW vs Skip-gram

- *TF-IDF:* TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a statistical measure used in natural language processing and information retrieval to evaluate the importance of a word within a document relative to a collection of documents or a corpus.
  1) **Term Frequency (TF)**
     This measures how frequently a term occurs in a document. It's calculated as the number of times a term appears in a document divided by the total number of terms in that document. Essentially, it reflects the importance of a term within the document itself.

$$\text{TF} = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

  2) **Inverse Document Frequency (IDF)** This measures how important a term is across the entire corpus. It's calculated as the logarithm of the total number of documents divided by the number of documents containing the term. Terms that appear in many documents will have a lower IDF, while terms that appear in fewer documents will have a higher IDF.

$$\text{IDF} = \log_e \left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t} \right)$$

  3) **TF-IDF Score**
     The TF-IDF score of a term in a document is the product of its TF and IDF scores. This means that the score increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

By calculating the TF-IDF scores for each term in a document, you can identify the terms that are most relevant to that document compared to the rest of the corpus. Terms with higher TF-IDF scores are considered more important or relevant to the document.

Example: TF-IDF Calculation Let's say we have the following three sentences as our corpus:

1) ”The cat sat on the mat.”
2) ”The dog played with the ball.”
3) ”The cat and the dog are friends.”

We’ll calculate the TF-IDF scores for each term in these sentences. For simplicity, we’re only considering unigrams (single words) and ignoring any punctuation.

Term Frequency (TF):
First, let’s calculate the term frequency (TF) for each term in each sentence:

Term	Sentence 1	Sentence 2	Sentence 3
the	2/6	1/6	2/7
cat	1/6	0	1/7
sat	1/6	0	0
on	1/6	0	0
mat	1/6	0	0
dog	0	1/6	1/7
played	0	1/6	0
with	0	1/6	0
ball	0	1/6	0
and	0	0	1/7
are	0	0	1/7
friends	0	0	1/7

Inverse Document Frequency (IDF):
Next, let’s calculate the inverse document frequency (IDF) for each term: Total number of documents (N) = 3

Token	IDF
the	0
cat	0.4055
sat	1.0986
on	1.0986
mat	1.0986
dog	0.4055
played	1.0986
with	1.0986
ball	1.0986
and	1.0986
are	1.0986
friends	1.0986

TF-IDF Score:
Finally, we calculate the TF-IDF score for each term in each sentence by multiplying the TF and IDF:

Token	Sentence 1	Sentence 2	Sentence 3
the	0	0	0
cat	0.0676	0	0.0579
sat	0.1831	0	0
on	0.1831	0	0
mat	0.1831	0	0
dog	0	0.0676	0.0579
played	0	0.1831	0
with	0	0.1831	0
ball	0	0.1831	0
and	0	0	0.1569
are	0	0	0.1569
friends	0	0	0.1569

.
This table shows the TF-IDF scores for each term in each sentence. Terms with higher TF-IDF scores are considered more important or relevant to the document because they are both frequent within the document and relatively rare across the corpus.

Observation: In natural language processing and information retrieval, when using the Term Frequency-Inverse Document Frequency (TF-IDF) representation for words in a document, it’s possible for certain words to have a TF-IDF value of 0. This can happen as the term not present in the document. If a specific term (word) is not present in the document, its term frequency (TF) will be 0. Consequently, when multiplied by the inverse document frequency (IDF), the overall TF-IDF value will be 0. This leads to many zero vectors when multiplied by Word2Vec vectors and thus we also observe only the **IDF** values as weights for Word2Vec vectors.

2) Bi-LSTM:
LSTM (Long Short Term Memory) is an improved version of a Recurrent Neural Network used for sequence and context analysis. This architecture has two separate LSTM networks, one gets the sequence of tokens as it is while the other gets in the reverse order. Both of these LSTM networks return a probability vector as output and the final output is the combination of both of these probabilities. It can be represented as:

$$p_t = p_{tf} + p_{tb}$$

$$p_{tf} = \text{Total vector probability from Forward LSTM}$$

$$p_{tb} = \text{Total vector probability from Backward LSTM}$$

Averaging word vectors leads to loss of sequential information, hence we move on to a Bi-LSTM based approach. While LSTMs are known to preserve the sequence information, Bi-directional LSTMs have an added advantage since they also capture both the left (past) and right (future) context in case of text classification. This helps the model to understand the sequential dependencies and patterns in the data more effectively.

To implement this, the Keras library is used. A RNN is created with 2 layers:

1) **Embedding Layer**: Accepts a one-hot representation of the sentences and returns a vector for each token.
2) **Bidirectional Layer**: Generates the vector embeddings.

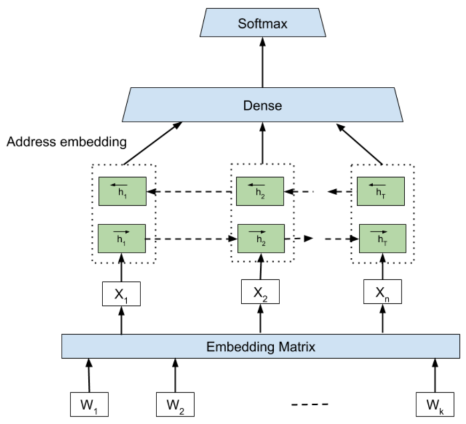
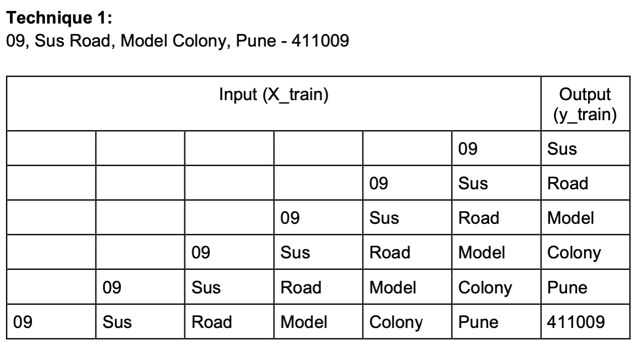


Fig. 2: Architecture of Bi-LSTM

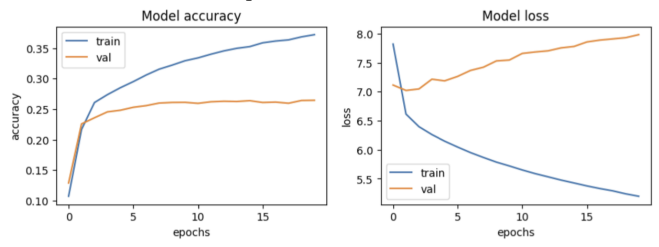
1) *Enhancements for Training Accuracy* To improve training accuracy, several enhancements were implemented:
   - **Word2Vec Initialization**: Instead of using randomly generated values, the embedding layer was initialized with the Word2Vec weight matrix. This approach leverages pre-trained word embeddings, which can capture semantic relationships between words and provide a better starting point for the model.
   - **Dropout Layers**: Dropout layers were incorporated to prevent overfitting by randomly dropping a fraction of units during training. This regularization technique helps to reduce the reliance of the model on specific features, leading to better generalization.
   - **L2 Regularization**: L2 regularization techniques were applied to the model's parameters. This penalizes large weights in the model, encouraging simpler models and reducing the risk of overfitting.
2) *Self-Supervised Learning for Language Modeling* Since the task of language modeling does not involve labeled data, self-supervised learning techniques were employed for training the model. Specifically, a self-supervised approach was adopted where the model learns to predict the next word in each sequence of text. By predicting the next word given a sequence of words, the model is effectively learning the underlying structure and patterns of the language. This enables the model to generate coherent and contextually relevant text. The following techniques were used to train the LSTM:



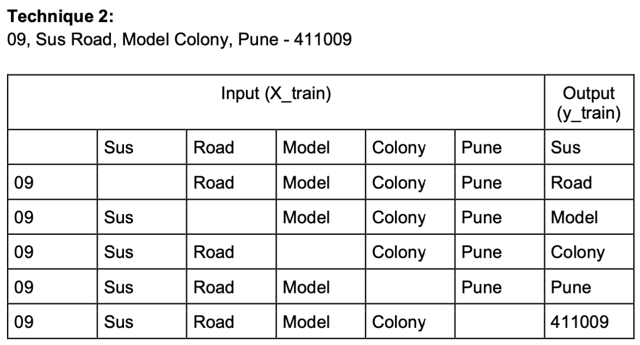
Technique 1:
09, Sus Road, Model Colony, Pune - 411009

Input (X_train)						Output (y_train)
					09	Sus
				09	Sus	Road
			09	Sus	Road	Model
		09	Sus	Road	Model	Colony
	09	Sus	Road	Model	Colony	Pune
09	Sus	Road	Model	Colony	Pune	411009

(a) Technique 1 - Next Word Prediction



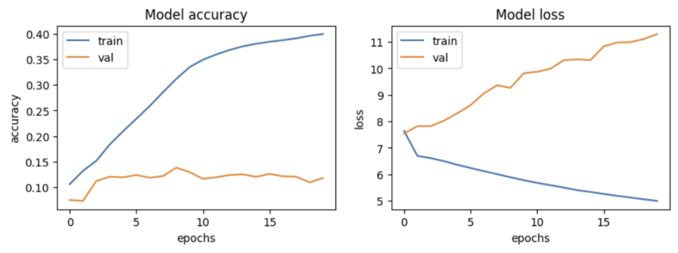
(b) Accuracy and Loss Graphs - Technique 1



Technique 2:
09, Sus Road, Model Colony, Pune - 411009

Input (X_train)						Output (y_train)
	Sus	Road	Model	Colony	Pune	Sus
09		Road	Model	Colony	Pune	Road
09	Sus		Model	Colony	Pune	Model
09	Sus	Road		Colony	Pune	Colony
09	Sus	Road	Model		Pune	Pune
09	Sus	Road	Model	Colony		411009

(a) Technique 2 - Missing Word Prediction



(b) Accuracy and Loss Graphs - Technique 2

3) RoBERTa:

The objective of using the RoBERTa model in address classification with Masked Language Modeling (MLM) is to enhance the model's ability to understand and predict contextual relationships within addresses. RoBERTa, by refining pre- training objectives, captures patterns and dependencies in the data. The MLM task involves masking parts of the input address and training the model to predict the masked components, enabling it to grasp address semantics. This approach aims to boost address classification accuracy by leveraging RoBERTa's contextual understanding, ultimately improving the model's capability to categorize diverse address formats with higher precision.

BERT based representations try to learn the context around a word and is able to better capture its meaning syntactically

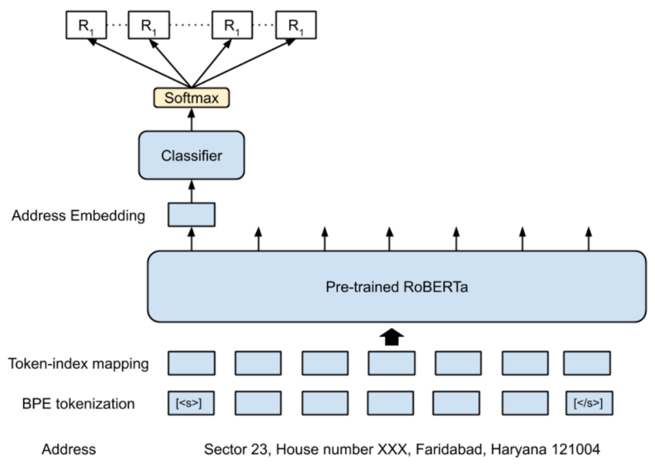


Fig. 5: RoBERTa-based Architecture for Classification

and semantically. For our context, NSP loss does not hold meaning since customer addresses on e-commerce platforms are logged independently. This motivates the choice of RoBERTa model since it uses only the MLM auxiliary task for pre-training over addresses. In experiments we use byte-level BPE tokenization for encoding addresses. We use perplexity score for evaluating the RoBERTa language model. It is defined as follows:

$$P(Sentence) = P(w1w2....wN)^{\frac{1}{N}}$$

where *P(Sentence)* denotes the probability of a test sentence and w1,w2,....,wN denotes words in the sentence. Generally, lower is the perplexity, better is the language model. After pre-training, the model would have learnt the syntactic and semantic aspects of tokens in shipping addresses. Figure 5 shows the overall approach used to pre-train RoBERTa model and fine-tune it for region classification.

IV. Experimental Setup

To test the models accurately and compare the working of the models discussed above, a synthetic dataset was curated which contained similar variations of some addresses along with different addresses. The models were assessed based on whether they could accurately predict the cosine similarity between the tokens.

The process of curating a synthetic dataset for testing the models involved several steps to ensure comprehensive evaluation and comparison. Here's an elaboration on each aspect of the dataset curation process:

1) **Synthetic Dataset Design and Composition**:
   - The synthetic dataset was designed to contain a diverse set of addresses, including variations of existing addresses and entirely new addresses.
   - Variations in the addresses included different spellings, abbreviations, variations in formatting, and other common variations found in real-world data.
   - The dataset also included addresses with similar components but different structures, such as addresses with the same street name but different house numbers or postal codes.
2) **Cosine similarity**: Cosine similarity is a metric used to measure the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the vectors, providing a value between -1 and 1, where:
   - 1 indicates that the vectors are perfectly aligned (i.e., pointing in the same direction),
   - 0 indicates that the vectors are orthogonal (i.e., perpendicular to each other),
   - -1 indicates that the vectors are perfectly anti-aligned (i.e., pointing in opposite directions).

   In the context of natural language processing or information retrieval, cosine similarity is often used to assess the similarity between documents or text passages represented as vectors in a high-dimensional space.
   The cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is calculated using the following formula:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

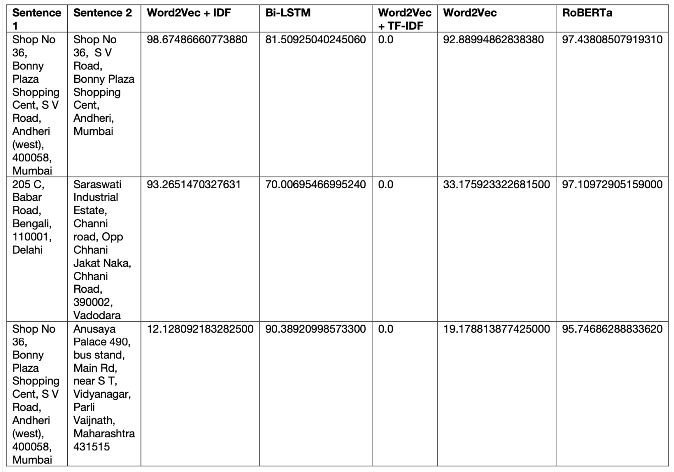
   Where:
   - $\mathbf{A} \cdot \mathbf{B}$ denotes the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$.
   - $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ denote the Euclidean norms (or magnitudes) of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively.

   Cosine similarity is commonly used in tasks such as document retrieval, text classification, and recommendation systems to measure the similarity between textual documents or items based on their content or features.
   - Cosine similarity was chosen as the metric for assessing the performance of the models.
   - For each pair of addresses in the dataset, the cosine similarity between their tokenized representations was calculated.
   - Cosine similarity provides a measure of similarity between two vectors, in this case, the tokenized representations of addresses.

Overall, the curated synthetic dataset provided a controlled environment for testing and comparing the performance of the models. By systematically varying the characteristics of the addresses and assessing their similarity using cosine similarity, the effectiveness of the models in handling variations and identifying similar addresses could be accurately evaluated.

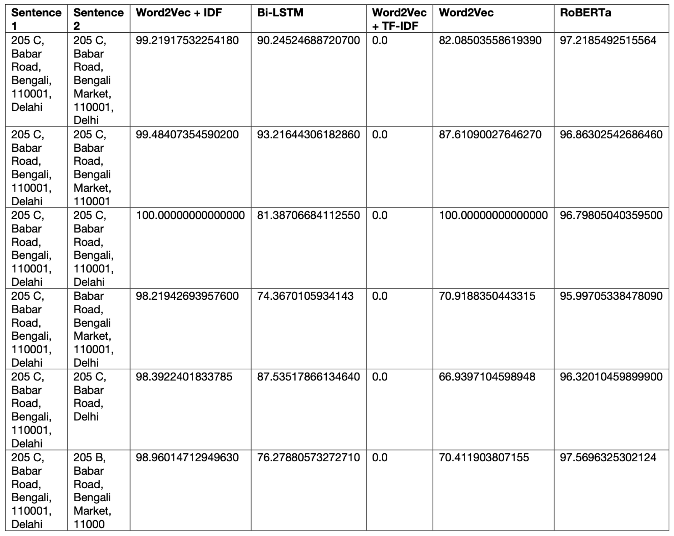
V. Conclusion and Future Work

In conclusion, this paper has addressed the intricate challenge of parsing customer addresses in the context of e-commerce, with a specific focus on the diverse landscape of Indian addresses. Through a multi-step methodology, we have demonstrated the efficacy of leveraging natural language processing (NLP) techniques to preprocess, standardize, and generate contextual embeddings for Indian addresses.



Sentence 1	Sentence 2	Word2Vec + IDF	Bi-LSTM	Word2Vec + TF-IDF	Word2Vec	RoBERTa
Shop No 36, Bonny Plaza Shopping Cent, S V Road, Andheri (west), 400058, Mumbai	Shop No 36, S V Road, Bonny Plaza Shopping Cent, Andheri, Mumbai	98.67486660773880	81.50925040245060	0.0	92.88994862838380	97.43808507919310
205 C, Babar Road, Bengali, 110001, Delahi	Saraswati Industrial Estate, Channi road, Opp Chhani Jakat Naka, Chhani Road, 390002, Vadodara	93.2651470327631	70.00695466995240	0.0	33.175923322681500	97.10972905159000
Shop No 36, Bonny Plaza Shopping Cent, S V Road, Andheri (west), 400058, Mumbai	Anusaya Palace 490, bus stand, Main Rd, near S T, Vidyanagar, Parli Vaijnath, Maharashtra 431515	12.128092183282500	90.38920998573300	0.0	19.178813877425000	95.74686288833620

Fig. 6: Analysis of all models - Different addresses



Sentence 1	Sentence 2	Word2Vec + IDF	Bi-LSTM	Word2Vec + TF-IDF	Word2Vec	RoBERTa
205 C, Babar Road, Bengali, 110001, Delahi	205 C, Babar Road, Bengali Market, 110001, Delhi	99.21917532254180	90.24524688720700	0.0	82.08503558619390	97.2185492515564
205 C, Babar Road, Bengali, 110001, Delahi	205 C, Babar Road, Bengali Market, 110001	99.48407354590200	93.21644306182860	0.0	87.61090027646270	96.86302542686460
205 C, Babar Road, Bengali, 110001, Delahi	205 C, Babar Road, Bengali, 110001, Delahi	100.00000000000000	81.38706684112550	0.0	100.00000000000000	96.79805040359500
205 C, Babar Road, Bengali, 110001, Delahi	Babar Road, Bengali Market, 110001, Delhi	98.21942693957600	74.3670105934143	0.0	70.9188350443315	95.99705338478090
205 C, Babar Road, Bengali, 110001, Delahi	205 C, Babar Road, Delhi	98.3922401833785	87.53517866134640	0.0	66.9397104598948	96.32010459899900
205 C, Babar Road, Bengali, 110001, Delahi	205 B, Babar Road, Bengali Market, 11000	98.96014712949630	76.27880573272710	0.0	70.411903807155	97.5696325302124

Fig. 7: Analysis of all models - Variations of similar addresses

We began by identifying common errors and variations in Indian addresses, acknowledging the lack of standardized formats and the presence of geographical ambiguity. Drawing inspiration from recent advancements in NLP, we proposed a systematic approach involving tokenization, custom token removal, lemmatization, and clustering of similar words using phonetic and string similarity algorithms.

To generate contextual vector embeddings, we explored several techniques including Word2Vec with TF-IDF, Bi-directional LSTM, and RoBERTa. Our experiments revealed that the Word2Vec along with IDF weight model and the pre-training RoBERTa on a large address dataset and fine-tuning it for classification tasks outperformed other methods, showcasing the ability to capture the complex semantics of Indian addresses.

Furthermore, we meticulously designed a synthetic dataset to comprehensively evaluate and compare the performance of the proposed methods. By calculating cosine similarity between tokenized representations of addresses, we accurately assessed the models' capability to handle variations and identify similar addresses.

Overall, this research represents a significant step towards enhancing the efficiency and accuracy of address parsing in the e-commerce domain, particularly in regions with non-standardized address formats like India. By leveraging state-of-the-art NLP techniques, we have laid the groundwork for future advancements in address processing, with potential applications ranging from supply chain management to fraud detection.

Moving forward, there are several avenues for further exploration, including experimenting with different tokenization strategies, pre-training variations of BERT models, and integrating additional contextual information to improve address classification accuracy. By continuing to innovate and refine these techniques, we can unlock new possibilities for enhancing the efficacy of e-commerce operations and customer experience.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR abs/1810.04805 (2018).

[2] Shreyas Mangalgi, Lakshya Kumar, Ravindra Babu, 2020. AI For Fashion Supply Chain, Aug 2020, San Diego, California - USA. Deep Contextual Embeddings for Address Classification in E-commerce.

[3] Vishal Kakkar, T. Ravindra Babu. SIGIR 2018 eCom, July 2018, Ann Arbor, Michigan, USA. Address Clustering for e-Commerce Applications.

[4] Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned Language Models for Text Classification.

[5] T. Ravindra Babu, Abhranil Chatterjee, Shivram Khandeparker, A. Vamsi Subhash, and Sawan Gupta. 2015. Geographical address classification without using geolo- cation coordinates. In Proceedings of the 9th Workshop on Geographic Information Retrieval, GIR 2015, Paris, France, November 26-27, 2015, Ross S. Purves and Christo- pher B. Jones (Eds.). ACM, 8:1–8:10.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.

[7] Vishal Kakkar and T. Ravindra Babu. 2018. Address Clustering for e-Commerce Applications. In eCOM@SIGIR.