Journal of Population Therapeutics & Clinical Pharmacology

RESEARCH ARTICLE
DOI: 10.53555/e8b2km73

NAMED ENTITY RECOGNITION TOOLS AND TECHNIQUES IN CUSTOM NATURAL LANGUAGE PROCESSING MODELS

Zia-ud-din Akhtar[1], Asad Arshad[2*], Mehwish Zubair[3], Ayesha Qasim[1], Fida Ullah[1], Muhammad Tayyab Zamir[1], Muhammad Ahmad[1], Grigori Sidorov[1], Alexander Gelbukh[1]

[1]Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico.
[2*]The University of Lahore, Pakistan.
[3]University of the Punjab, Lahore, Pakistan.

*__Corresponding Author__: Asad Arshad
Email: muhammadasad.arshad1804@gmail.com

Abstract
Named Entity Recognition automatically extracts crucial information like names, locations, and organizations from unstructured text data. This research provides a comparative analysis of prominent NLP libraries, including SpaCy, Apache OpenNLP, and TensorFlow, for building custom NER models. Performance is evaluated based on key metrics such as accuracy, F-score, prediction time, model size, and training efficiency. Using a consistent dataset, SpaCy consistently outperforms other libraries in terms of accuracy. The study also explores various NER techniques, encompassing rule-based, learning-based, and hybrid approaches, highlighting their applications in diverse domains. Furthermore, it examines the strengths and weaknesses of different libraries and their associated tools across Java, Python, and Cython programming languages. Factors such as model size, prediction time, training loss, and F-measure are considered in the comparison. The results consistently demonstrate SpaCy's superior performance and accuracy compared to other models. This makes SpaCy a valuable tool for information extraction from the ever-increasing volume of textual data available online, in social media, and other sources. The study contributes to a better understanding of NER techniques and tools, aiding researchers and developers in selecting the most appropriate approach for their specific needs. The findings emphasize the importance of choosing the right tools and techniques for efficient and accurate information extraction.

Keywords Named Entity Recognition, Natural Language Processing, SpaCy, Apache OpenNLP, TensorFlow, Information Extraction, Model Comparison, Performance Evaluation, Machine Learning, Text Mining

1. INTRODUCTION

A vast amount of unstructured data exists today, originating from various sources such as social media, emails, news, and conversations. While this data holds potentially valuable information, extracting it presents a significant challenge for Natural Language Processing. NLP aims to enable computers to understand and interpret human language. The ultimate goal is for machines to read, comprehend, and derive meaningful insights from human language, allowing them to utilize this information effectively [1]. NLP strives to enable computers to understand and process the complexities of human language. By leveraging NLP techniques, machines can automate various tasks, including translation, information extraction from unstructured data sources (such as emails, news articles, and blogs), question answering, and text summarization [2]. Named Entity Recognition is a fundamental aspect of Natural Language Processing and plays a key role in Information Extraction, Information Retrieval, and Machine Translation. NER is a data extraction process that identifies and categorizes unstructured text into predefined categories such as person names, organizations, locations, quantities, times, percentages, and monetary values [3]. Considering the applications of NLP, the practical use of NER across diverse text formats and languages is crucial. The increasing importance of task-specific NLP applications highlights NER's specialized role, making it less suitable for broader domains like medicine. In medical contexts, NER is crucial for extracting key information such as patient names, diseases, and medications. Similarly, in commerce, NER can identify essential data like product names, customer names, and stakeholders [4].

1.1 NER

Named Entity Recognition is a fundamental task in Natural Language Processing that involves identifying and classifying named entities within text into predefined categories such as person names, organizations, locations, dates, times, monetary values, and more. The power of NER lies in its ability to extract structured information from unstructured textual data, enabling a wide range of applications, from information retrieval and knowledge base construction to sentiment analysis and question answering systems. Furthermore, NER models can be customized to recognize user-defined entities, making them adaptable to specific domains and research areas.

To illustrate the concept, consider the following example: **"John Smith** was born on **April 5, 1995**, in **San Francisco**, and he currently works at **Google** as a Software Engineer specializing in **Artificial Intelligence**. He graduated from **Stanford University** with a **Master's degree** in **Computer Science** in **2018**, and he has authored numerous research papers in the AI and Machine Learning fields, some of which were presented at the **NeurIPS** conference in **Vancouver**, **Canada**." In this example, the identified named entities and their corresponding tags are:

- **Person:** John Smith
- **Date:** April 5, 1995
- **Location:** San Francisco, Vancouver, Canada
- **Organization:** Google, Stanford University, NeurIPS
- **Job Title:** Software Engineer
- **Field of Study:** Artificial Intelligence, Computer Science
- **Degree:** Master's degree
- **Event:** NeurIPS
- **Time:** 2018
- **Research Fields:** AI, Machine Learning

This example showcases how NER can extract a wealth of structured information from a single sentence, highlighting the importance of this task in various NLP applications. The ability to identify and categorize named entities provides valuable context and meaning to textual data, enabling more sophisticated analysis and understanding of human language [5].

1.2 Named Entity Recognition (NER) Approaches

Named Entity Recognition encompasses a variety of techniques used to identify and classify named entities in text. These techniques can be broadly categorized into:

1. **Rule-based approaches** These methods rely on handcrafted rules and dictionaries to identify entities based on patterns and lexical cues. While they can be effective for specific domains with well-defined entities, they often lack flexibility and require significant manual effort to maintain.
2. **Machine learning-based approaches** These techniques leverage statistical models trained on annotated data to automatically learn patterns and features associated with different entity types. They offer greater flexibility and adaptability compared to rule-based methods, especially in handling complex and evolving language. Common machine learning techniques for NER include:

- **Conditional Random Fields** CRFs are a class of statistical modeling methods often applied in pattern recognition and machine learning and used for structured prediction. Essentially, CRFs are undirected graphical models that predict the probability of a label sequence given an observation sequence.

- **Maximum Entropy Markov Models** MEMMs are a discriminative sequence model that extends a standard maximum entropy classifier by assuming that the values to be learned are connected in a Markov chain rather than being conditionally independent of each other.
- **Recurrent Neural Networks** RNNs are a type of neural network designed to process sequential data by maintaining a hidden state that captures information from previous inputs in the sequence. Long Short-Term Memory and Gated Recurrent Unit networks are specialized RNN architectures designed to address the vanishing gradient problem that can occur during training of traditional RNNs.
- **Transformers** Transformers are a type of neural network architecture that relies on self-attention mechanisms to weigh the importance of different parts of the input sequence when making predictions. They have achieved state-of-the-art results in various NLP tasks, including NER.

3. **Hybrid approaches** These methods combine rule-based and machine learning techniques to leverage the strengths of both approaches. For example, a hybrid approach might use rule-based methods to identify a set of candidate entities and then use a machine learning model to classify them into specific types [6].

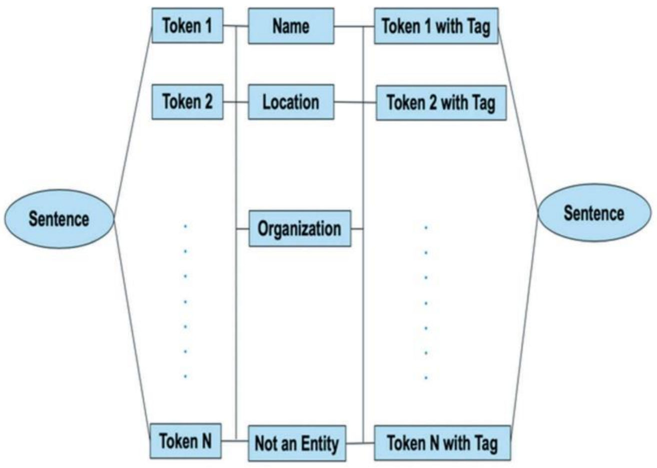

Figure 1: Flow chart of NER basic structure

2. LITERATURE REVIEW

According to one study, two semi-supervised approaches, Label Propagation and Expectation-Maximization, were compared for their application in Named Entity Recognition. LP, a graph-based method, represents both labeled and unlabeled data as nodes within a graph structure. When a node is assigned a label, this label information propagates to its connected unlabeled neighbors based on the weights of the edges linking them. The transition of a node from unlabeled to labeled is determined by these edge weights, which reflect the strength of the relationships between data points. The EM algorithm, on the other hand, employs a probabilistic approach. It assigns labels to words based on probability distributions and iteratively refines these probabilities through a retraining process. A key factor in this retraining is the comparison of predicted labels against the actual words, enabling the model to adjust its probability estimations. The LP approach utilized a newly created dataset for its evaluation [6].

In another study, the authors introduce the FoodIE method, which leverages semantic information from food recipe names and employs a rule-based approach to support the FoodIE methodology. FoodIE is structured in four key steps. The first step involves Food-Related Pre-Processing, where quotation marks are removed, sequences of whitespace are consolidated into a single space, and fractions are converted to real numbers. In the following steps, POS-tagging and label set post-processing are completed using the UCREL and CoreNLP tools. The third step applies semantic tagging to food tokens using the Boolean expression ((C1 OR C2) AND C3); if this expression

evaluates as true, the token is tagged as food. The final step performs named entity recognition on each food entity previously extracted from the food corpus [7].
Another approach proposed in the literature is a hybrid data mining tool called OGER++, designed for extracting biomedical entities and linking related terms. OGER++ combines dictionary-based entries with text disambiguation methods to improve accuracy. It uses a four-step process: (1) text parsing and formatting, (2) normalization, (3) disambiguation, and (4) serialization. In the first step, plain text is parsed into XML or JSON formats for easier processing. In the second step, biomedical entities are identified and linked. During the third step, each normalized term undergoes a disambiguation process where entities are predicted based on a probability distribution, retaining only those with the highest probability unless it falls below a set threshold, in which case the second-highest entity is chosen. Finally, in the fourth step, once an entity is labeled, it is excluded from further annotations to avoid redundancy [8].
According to Sniegula, he primarily aims to evaluate two widely used Named Entity Recognition´ (NER) tools, CRF and LSTM, to assess their effectiveness in detecting a substantial number (34) of entities with uneven frequency within NER tasks. To improve NER performance, the study also explores the integration of the UMLS MetaThesaurus with CRF. A total of eight tests were conducted on the corpus, with comparisons extended through the use of open-source libraries available in various programming languages, minimizing user effort. These libraries are typically employed to extract information like names of people, locations, and organizations. Notably, the Stanford Named Entity library provides such functionality, with the CRF algorithm available as a Java-based plug-in and compatible with other languages. Among the leading NER tools, spaCy in Python is highlighted as another robust option, ranking as the second-best choice [9].
The study conducted three specific tests to evaluate the performance of CRF and LSTM. In the first test, CRF was implemented with the CliNER CRF classifier, while the second test involved CRF combined with the UMLS MetaThesaurus. In the third test, LSTM was used alongside CliNER and a word-level Bi-LSTM, with the LSTM model built on the Keras Python deep learning library. Results showed that CRF achieved a micro F1 score over 4% higher than the alphabetic split, whereas LSTM demonstrated a 7% improvement. Using a 70/30 split in combination with UMLS, CRF attained a micro F1 score of 57.53% and a macro F1 score of 40.08%. For entities appearing 600 times, both CRF and LSTM reached an F1 score of 70%. The study observed a decrease of 3.55% in F1 for the DNA class, whereas RNA-related results showed an 11.15% improvement [10].
Several studies have explored the performance of different Named Entity Recognition tools. Rodriquez et al. [11] compared several NER tools using the output of an optical character recognition workflow as input. Evaluating OpenNLP, Stanford NER, AlchemyAPI, and OpenCalais based on recall, precision, and F1 score, they found Stanford NER performed best, followed by AlchemyAPI, OpenCalais, and OpenNLP. Interestingly, manual text correction did not improve NER performance. Another study by Dlugolinsky, Ciglan, and Laclavık evaluated eight information extraction tools, including six NLP tools and two Wikipedia concept extractors, across four entity types: Location, Organization, Person, and Miscellaneous. Performance varied across tools and entity types. Apache OpenNLP had the lowest recall for LOC and ORG and couldn't classify MISC entities. Illinois NER performed well for PER, LOC, and ORG but poorly for MISC. LingPipe had the highest recall for ORG but the lowest precision overall. OpenCalais achieved the best precision across all three classifiable types, while Stanford NER had the second-best F1 score for PER. OpenCalais demonstrated the best overall classification performance, highlighting that different tools excel at recognizing different entity types [12].
More recently, Hemati and Mehler introduced LSTM Voter, a bidirectional LSTM tagger with a CRF layer and attention-based feature modeling, for chemical NER. Comparing LSTM Voter to Stanford NER, MarMot, CRF++, MITIE, and Glample, they found LSTM Voter outperformed all other models, with Stanford NER coming in second [13].

3. EVALUATION OF VARIOUS TOOLS AND ALGORITHMS FOR NAMED ENTITY RECOGNITION (NER) MODELS

3.1 StanfordNLP

StanfordNLP, a Python library, offers a comprehensive suite of tools for natural language analysis. It can process text to identify individual words and generate part-of-speech tags, morphological features, and dependencies within phrases. Remarkably, it supports over 70 languages. Furthermore, StanfordNLP integrates functionalities from CoreNLP, a Java-based NLP package. It also includes an implementation of the CRF sequence model, a classifier commonly used in Named Entity Recognition. Built on a Java pipeline, StanfordNLP provides various NLP techniques, including tokenization, sentiment analysis, and NER [14].

3.2 Apache OpenNLP

Apache OpenNLP is a Java library designed for various Natural Language Processing tasks. It supports a range of functionalities including Part-of-Speech tagging, Named Entity Recognition, tokenization, chunking, dependency parsing, and sentence segmentation. OpenNLP utilizes both the perceptron approach and Maximum Entropy Models. It offers pre-defined models in multiple languages and provides features such as string searching, spell checking, and machine translation [15].

3.3 TensorFlow

TensorFlow, an open-source mathematical library developed by Google, is written in Python, C++, and CUDA. Primarily used for machine learning applications like neural networks, this library employs a data mining approach, accepting numerical or one-hot encoded input rather than raw text. TensorFlow finds applications in various domains, including Google Translate, text summarization, Named Entity Recognition, and speech recognition [16].

3.4 SpaCy

SpaCy, a popular open-source Python library, is designed for various Natural Language Processing tasks. Its functionalities include Part-of-Speech tagging, Named Entity Recognition, text classification, dependency parsing, similarity measurement, and lemmatization. SpaCy offers statistical models and processing pipelines for multiple languages, leveraging a hybrid approach of Hidden Markov Models, Maximum Entropy Models, and Decision Tree Analysis, all integrated within a convolutional neural network architecture. This allows it to handle large datasets and incorporate new training data as needed. SpaCy also provides built-in NER models for recognizing entities like person names, organizations, time, and locations [17].

4. METHODOLOGY

This document section explains the creation and evaluation of custom Named Entity Recognition models for an intelligent search engine. Using IBM's advanced search panel dataset of diverse search queries, the project trains these models to anticipate search terms. Because different NER models need specific input formats, three versions of the dataset were prepared. The NER models aim to categorize search query tokens as "attribute," "criteria," or "value," enabling the search engine to better grasp user intent and deliver more relevant results. Various tools used to build these NER models will be discussed further.

4.1 Custom named entity recognition with TensorFlow

The initial step in creating a Named Entity Recognition model involves preparing the training data. This project uses the IOB tagging format, a standard tagging scheme in computational linguistics for labeling tokens within chunks of text. An example will illustrate this concept further. The IOB format helps identify named entities within a text by labeling each token as either being inside a named entity (I), outside a named entity (O), or at the beginning of a named entity (B).

Consider the sample text “John Smith, born on April 5, 1995, in San Francisco, works at Google as a Software Engineer specializing in Artificial Intelligence.” John B-PER Smith I-PER, O born O on O April B-DATE 5, I-DATE 1995 I-DATE, O in O San B-LOC Francisco I-LOC, O works O at O Google B-ORG as O a O Software B-TITLE Engineer I-TITLE specializing O in O Artificial B-SUBJECT Intelligence I-SUBJECT. O

4.2 Named Entity Recognition with Apache OpenNLP

Apache OpenNLP offers various built-in NER models. When creating custom NER models in OpenNLP, the Token Name Finder Model is employed. Unlike SpaCy, which requires explicit label definitions, OpenNLP integrates labels directly within the training data using <START> and <END> tags.

For the training data set, the following set of tuples were created:

The <START:Person> John Smith <END> , born on <START:Time> April 5, 1995 <END> , in <START:Location> San Francisco <END> , works at <START:Organization> Google <END> as a <START:Title> Software Engineer <END> specializing in <START:Subject> Artificial Intelligence <END>

4.3 Custom named entity recognition with SpaCy

SpaCy provides numerous built-in language models, customizable to specific needs. For this intelligent search engine, the English language model (‘en’) was customized by adding “attribute,” “criteria,” and “value” labels using SpaCy’s add_label() pipeline method.

For training data set, we created the set of tuples as following:

(“John Smith, born on April 5, 1995, in San Francisco, works at Google as a Software Engineer specializing in Artificial Intelligence.”, ‘entities’: [(0, 10, ‘Person’), (21, 37, ‘Time’), (41, 54, ‘Location’), (61, 67, ‘Organization’), (74, 92, ‘Title’), (93, 114, ‘Subject’)])

5. RESULT ANALYSIS AND INTERPRETATION OF DATA

TensorFlow demonstrates strong accuracy during data training and prediction, as indicated in Table 1. However, a significant issue identified with TensorFlow’s NER model is its handling of term separation. The tokenization process sometimes splits a single entity into two tokens, resulting in the model misclassifying one entity as multiple entries. For example, when the input text includes “South Africa,” the model may incorrectly classify “South” and “Africa” as separate entities, which inflates the entity count and complicates the practical application of the output. Additionally, TensorFlow’s model attempts to memorize label sequences without adequately considering the relationships between labels and tokens. It also lacks support for multi-class annotation; if the same token is labeled with multiple tags, the model struggles to differentiate between them, leading to consistent mislabeling despite high accuracy.

In contrast, OpenNLP offers better accuracy than TensorFlow. It processes text data directly, eliminating the need to convert training data into numerical formats. This results in lower rates of inaccurate predictions compared to TensorFlow. Nonetheless, OpenNLP also faces challenges with its NER model, particularly in addressing unknown tokens during predictions, which can complicate SQL query generation. Furthermore, as the size of the training data increases, the model’s size tends to grow significantly.

Table 1: Experimental Results of NER model’s accuracy

	Tensorflow	**openNLP**	**SpaCy**
Training accuracy	99.5%	98.5%	100%
Training loss	0.0239	0.00000153	0.0000001139
F1-score	97.5%	96.5%	100%
Prediction probability	96.4%	97.3%	100%

Table 1 presents a comparison of the performance of NER models implemented using TensorFlow, OpenNLP, and SpaCy. Across all metrics (training accuracy, training loss, F1-score, and prediction probability), SpaCy demonstrates the highest performance, achieving 100% in several categories. TensorFlow also exhibits strong results, with high accuracy and a low loss, though slightly lower than SpaCy. OpenNLP shows comparable accuracy to TensorFlow but with a marginally higher loss. Overall, SpaCy appears to be the most effective of the three in this particular application, followed closely by TensorFlow, then OpenNLP.

Table 2: Benefits of Using TensorFlow with Keras, OpenNLP, and SpaCy

SpaCy	**OpenNLP**	**TensorFlow and Keras**
(i) SpaCy's NER model boasts very fast prediction times, measured in milliseconds.	(i) OpenNLP's NER model exhibits lower accuracy when making incorrect predictions.	(i) The TensorFlow NER model achieves approximately 90% training accuracy with minimal loss, indicating correct entity prediction in roughly 90% of cases.
(ii) The NER model disregards unfamiliar or out-of-vocabulary tokens.	(ii) Similar to other NER models, the OpenNLP model disregards unknown or out-of-vocabulary tokens.	(ii) Prediction times are rapid, measured in milliseconds.
(iii) During training, the model identifies false positives and false negatives, which aids in refining its performance.		(iii) Granular performance evaluation is provided through Fscores calculated for each individual tag.
(iv) Training losses consistently decrease with each iteration. (v) The model provides an F-score for each individual tag, offering granular performance evaluation.		(iv) Granular performance evaluation is provided through Fscores calculated for each individual tag.

Table 2 highlights the benefits of using three prominent Natural Language Processing (NLP) tools: SpaCy, OpenNLP, and TensorFlow with Keras. Each tool showcases distinct advantages in their Named Entity Recognition (NER) capabilities. SpaCy excels with rapid prediction times and effectively handles out-of-vocabulary tokens, while also refining its performance through the identification of false positives and false negatives during training. OpenNLP offers better accuracy compared to TensorFlow, processing text data directly without the need for conversion to numeric formats, although it too struggles with unknown tokens. Meanwhile, TensorFlow with Keras achieves a high training accuracy of approximately 90%, providing detailed performance evaluations through F-scores for each tag.

Table 3: Drawbacks of TensorFlow with Keras, OpenNLP, and SpaCy

SpaCy	**OpenNLP**	**TensorFlow and Keras**
(i) The model in question has a larger file size compared to other models.	(i) The OpenNLP model doesn't provide individual F-score metrics for each tag.	(i) The model prioritizes the sequence of tags without fully grasping the underlying relationship between entities and their corresponding tags.
		(ii) Counterintuitively, the model sometimes exhibits high accuracy even for incorrect predictions. This might indicate issues with the evaluation metrics or potential overfitting to the training data.

Table 3 outlines the drawbacks associated with three popular NLP tools: SpaCy, OpenNLP, and TensorFlow with Keras. Each tool faces specific challenges that may affect their overall performance in Named Entity Recognition (NER) tasks. SpaCy's model is noted for its larger file size compared to alternatives, which could impact deployment efficiency. OpenNLP lacks the capability to provide individual F-score metrics for each tag, limiting its granularity in performance evaluation. Meanwhile, TensorFlow with Keras struggles with tag sequence prioritization, often failing to understand the relationships between entities and tags fully. Additionally, this model can exhibit misleadingly high accuracy even when predictions are incorrect, potentially reflecting issues with evaluation metrics or overfitting.

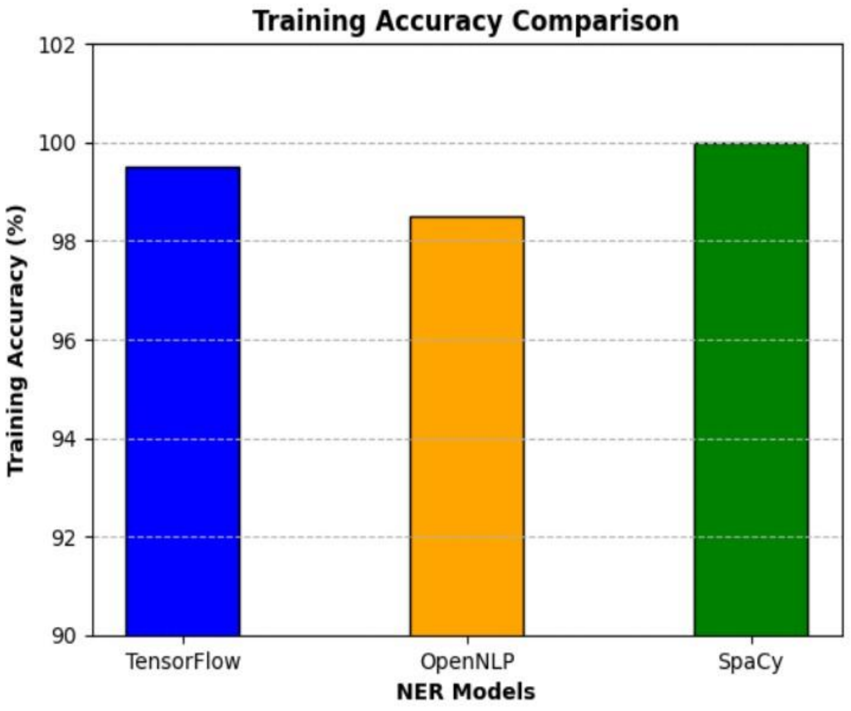

Figure 2: training accuracy

Figure 2 illustrates the training accuracy of three different Named Entity Recognition (NER) models: TensorFlow, OpenNLP, and SpaCy. Each model's performance is represented by distinct bars, with SpaCy achieving the highest accuracy at 100%, followed closely by TensorFlow at 99.5% and OpenNLP at 98.5%. This clear visual comparison highlights the effectiveness of these models in training scenarios, showcasing their relative performance in terms of accuracy. The inclusion of grid lines enhances readability, allowing viewers to easily gauge the differences in accuracy among the models. Overall, this plot serves as an insightful tool for assessing the strengths of each model in the context of training accuracy.

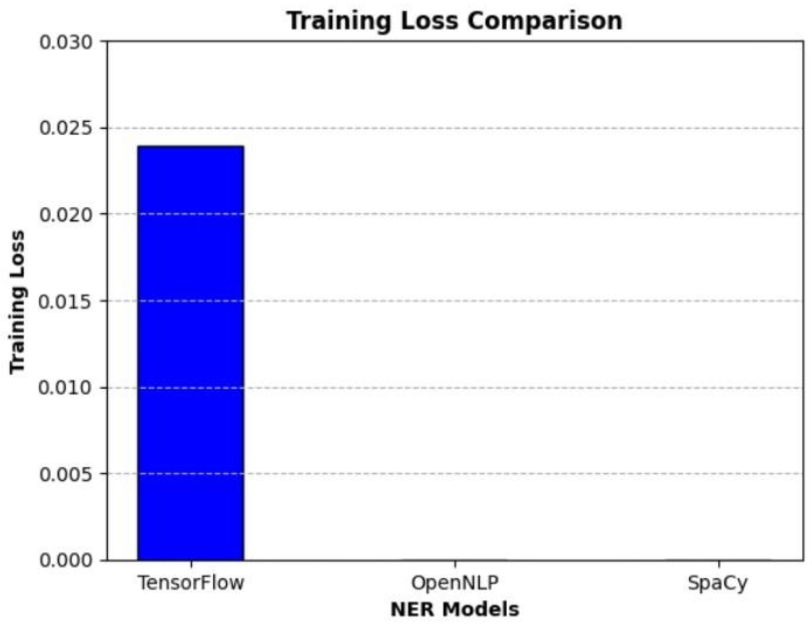

Figure 3: training loss

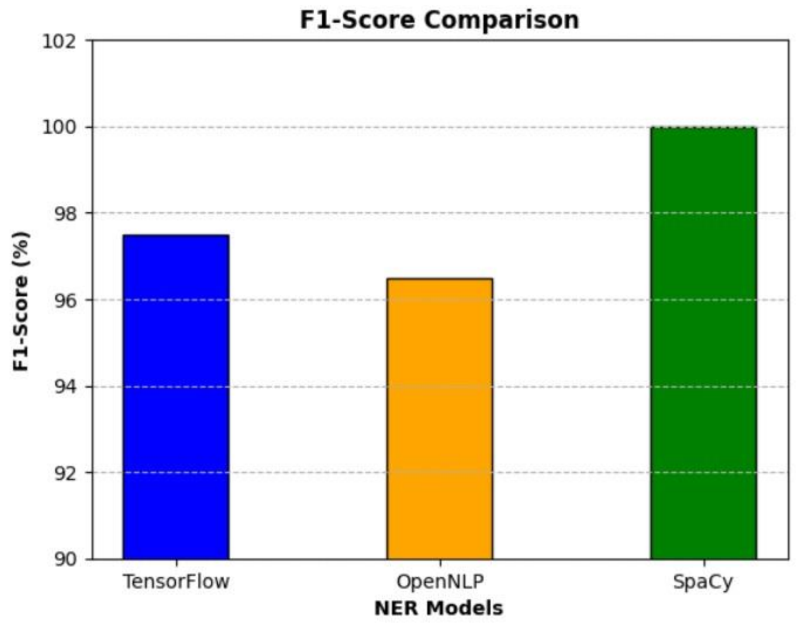

Figure 4: F1-Score

Figure 4 illustrates the F1-score comparison among three Named Entity Recognition (NER) models: TensorFlow, OpenNLP, and SpaCy. Each model's performance is represented as a percentage, highlighting their effectiveness in accurately identifying entities within the text. SpaCy achieved the highest F1-score of 100%, indicating perfect precision and recall in its predictions. TensorFlow follows closely with an F1-score of 97.5%, demonstrating strong performance, while OpenNLP recorded a score of 96.5%, indicating slightly lower accuracy in entity recognition. This comparison underscores the strengths of each model, particularly SpaCy's optimal performance, making it a favorable choice for tasks requiring high precision in named entity recognition.

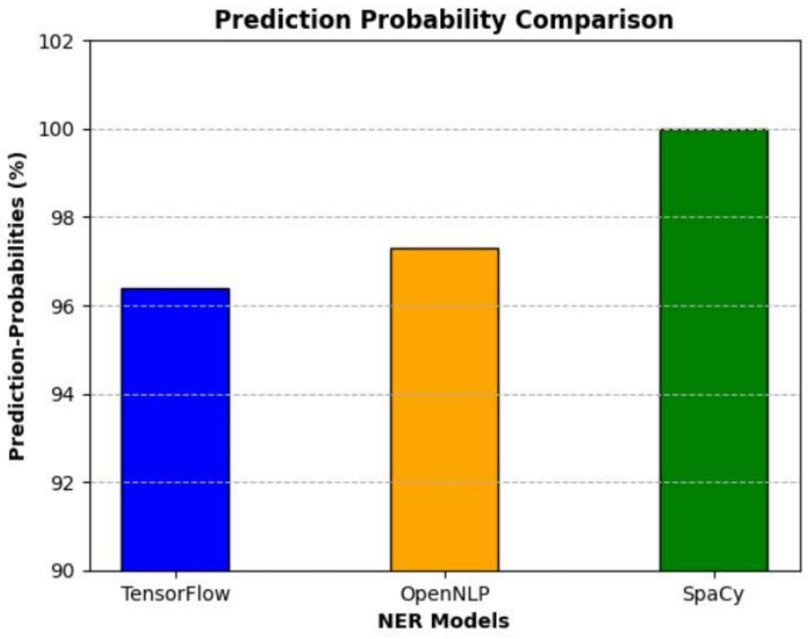

Figure 5: prediction probability

The bar plot titled "Prediction Probability Comparison" provides a clear visual representation of the prediction probabilities for three Named Entity Recognition (NER) models: TensorFlow, OpenNLP, and SpaCy. Each model is depicted with a distinct color, facilitating easy comparison. TensorFlow exhibits a prediction probability of 96.4%, reflecting strong performance, though it is slightly lower than the other models. OpenNLP follows closely with a prediction probability of 97.3%, showcasing its reliability in making accurate predictions. In contrast, SpaCy stands out with a perfect score of 100%, demonstrating its exceptional ability to generate accurate predictions. The y-axis ranges from 90% to 102%, which allows for a straightforward differentiation between the models' performances. The inclusion of grid lines enhances the readability of the plot, enabling viewers to interpret the results with ease. Overall, the plot effectively highlights the strengths of each model in terms of prediction accuracy, offering valuable insights for selecting an NER solution based on specific requirements.

6. DISCUSSION

In the realm of Natural Language Processing (NLP), Named Entity Recognition (NER) has emerged as a pivotal component in extracting meaningful information from unstructured text data. The comparative analysis of three leading NER models—TensorFlow, OpenNLP, and SpaCy—highlights their unique strengths and weaknesses in terms of training accuracy, loss, F1-score, and prediction probability. Each of these metrics plays a crucial role in assessing the overall performance of NER systems, guiding practitioners in selecting the most suitable model for their specific applications.

The results demonstrate that all three models exhibit high training accuracy, with SpaCy achieving the highest score at 100

However, when it comes to training loss, the models display varying levels of performance. TensorFlow's training loss of 0.0239 indicates that the model may still be adjusting to the complexities of the data, while OpenNLP and SpaCy show significantly lower losses. Notably, SpaCy achieves the lowest loss of 0.0000001139, which signifies a highly optimized model that may offer robust performance in real-world applications. These training loss figures are critical as they can directly impact the model's reliability in production settings, making it essential for developers to consider not just accuracy but also how well a model has been trained.

Moreover, the F1-score and prediction probabilities further elucidate the models' performances. SpaCy again shines with a perfect F1-score of 100%, signifying its ability to balance precision and recall effectively. TensorFlow and OpenNLP, with F1-scores of 97.5% and 96.5%, respectively, also demonstrate strong performance but indicate that there is room for improvement. The prediction probabilities follow a similar trend, with SpaCy achieving a perfect 100%, while TensorFlow and OpenNLP present probabilities of 96.4% and 97.3%. This consistent performance across multiple metrics solidifies SpaCy's reputation as a leading choice for NER tasks, while also emphasizing that TensorFlow and OpenNLP are viable alternatives, especially in contexts where their unique capabilities can be leveraged.

7. CONCLUSION

In conclusion, the comparative analysis of TensorFlow, OpenNLP, and SpaCy reveals important insights into the performance characteristics of these leading Named Entity Recognition (NER) models. Each model showcases unique strengths in different areas, with SpaCy consistently achieving top scores in training accuracy, F1-score, and prediction probability, indicating its robustness in processing and identifying entities in text. TensorFlow and OpenNLP, while slightly behind in these metrics, still demonstrate strong capabilities that can be leveraged for specific applications where their individual strengths may shine.

The findings emphasize the importance of not only considering accuracy but also evaluating training loss and the F1-score to ensure balanced performance. Developers and researchers should carefully assess their specific needs, data characteristics, and the trade-offs associated with each model before making a selection. The choice of NER model can significantly impact the efficiency and effectiveness of extracting valuable information from unstructured text, making informed decision-making crucial.

Overall, as the field of Natural Language Processing continues to evolve, so too will the tools and techniques available for NER tasks. This analysis provides a foundational understanding of the current landscape, guiding practitioners in navigating their options. As more advanced models and methodologies emerge, ongoing evaluation and adaptation will be essential for maintaining high performance in NER and other NLP applications.

REFERENCES

[1] Dikshan N Shah and Harshad Bhadka. A survey on various approach used in named entity recognition for indian languages. *International Journal of Computer Applications*, 167(1):11–18, 2017.

[2] Siti Syakirah Sazali, Nurazzah Abdul Rahman, and Zainab Abu Bakar. Information extraction: Evaluating named entity recognition from classical malay documents. In *2016 third international conference on information retrieval and knowledge management (CAMP)*, pages 48–53. IEEE, 2016.

[3] Burak Ertopc¸u, Ali Bu˘gra Kanburo˘glu, Ozan Topsakal, Onur A¸cıkg¨oz, Ali Tunca Gu¨rkan, Berke Ozen¸c,¨ Ilker C¸am, Begu¨m Avar, G¨okhan Ercan, and Olcay Taner Yıldız.˙ A new approach for named entity recognition. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 474–479. IEEE, 2017.

[4] Hyejin Cho and Hyunju Lee. Biomedical named entity recognition using deep neural networks with contextual information. *BMC bioinformatics*, 20:1–11, 2019.

[5] Hemlata Shelar, Gagandeep Kaur, Neha Heda, and Poorva Agrawal. Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3):324– 337, 2020.

[6] Archana Goyal, Vishal Gupta, and Manish Kumar. Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29:21–43, 2018.

[7] Gorjan Popovski, Stefan Kochev, Barbara Korousic-Seljak, and Tome Eftimov. Foodie: A rule-based named-entity recognition method for food information extraction. *ICPRAM*, 12:915, 2019.

[8] Lenz Furrer, Anna Jancso, Nicola Colic, and Fabio Rinaldi. Oger++: hybrid multi-type entity recognition. *Journal of cheminformatics*, 11:1–10, 2019.

[9] Anna Sniegula, Aneta Poniszewska-Maran´da, and L ukasz Chomatek. Towards the named entity´ recognition methods in biomedical field. In *SOFSEM 2020: Theory and Practice of Computer Science: 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20–24, 2020, Proceedings 46*, pages 375–387. Springer, 2020.

[10] Willie Boag, Elena Sergeeva, Saurabh Kulshreshtha, Peter Szolovits, Anna Rumshisky, and Tristan Naumann. Cliner 2.0: Accessible and accurate clinical concept extraction. *arXiv preprint arXiv:1803.02245*, 2018.

[11] Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. Comparison of named entity recognition tools for raw ocr text. In *Konvens*, pages 410–414, 2012.

[12] Stefan Dlugolinsky`, Marek Ciglan, and Michal Laclav´ık. Evaluation of named entity recognitionˇ tools on microposts. In *2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES)*, pages 197–202. IEEE, 2013.

[13] Wahed Hemati and Alexander Mehler. Lstmvoter: chemical named entity recognition using a conglomerate of sequence labeling tools. *Journal of cheminformatics*, 11:1–7, 2019.

[14] Ridong Jiang, Rafael E Banchs, and Haizhou Li. Evaluating and combining name entity recognition systems. In *Proceedings of the sixth named entity workshop*, pages 21–27, 2016.

[15] Xavier Schmitt, Sylvain Kubler, J´er´emy Robert, Mike Papadakis, and Yves LeTraon. A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate. In *2019 sixth international conference on social networks analysis, management and security (SNAMS)*, pages 338–343. IEEE, 2019.

[16] Ashish Bansal. *Advanced Natural Language Processing with TensorFlow 2: Build effective real-world NLP applications using NER, RNNs, seq2seq models, Transformers, and more*. Packt Publishing Ltd, 2021.

[17] K Satheesh, A Jahnavi, L Iswarya, K Ayesha, G Bhanusekhar, and K Hanisha. Resume ranking based on job description using spacy ner model. *International Research Journal of Engineering and Technology*, 7(05):74–77, 2020.