Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 45 No. 6
June 2024

Deep Analysis and Advancemts in Named Entity Recognition - NER

[1]Madhu N, [2]Dr.Veenadhari Suraparaju,[3]Dr.Shivamurthaiah M
[1]Researcher in Data Science,[2]Dean(CS/IT) at RNTU,[3]Associate Professor at BIT
[1]Computer Science & Applications,
[1]Rabindranath Tagore University , Madhya Pradesh, Bhopal

Abstract
Named Entity Recognition (NER) is an essential task in natural language processing (NLP) that entails locating and categorizing named entities in unstructured text, including names of people, places, organizations, dates, and more. There is a rising need for precise and effective NER systems because to the volume and complexity of textual material that is available online. This study offers a thorough examination and current state-of-the-art review of Named Entity Recognition.
The first section of the paper introduces the basic ideas of NER and discusses its use in a number of NLP applications, including sentiment analysis, information retrieval, and question answering. After that, it explores the conventional NER techniques, such as statistical models and rule-based systems, outlining the benefits and drawbacks of each[1][2]. The study then delves into the development of deep learning methods in NER and their revolutionary influence on the area. By extracting contextual information and intricate linguistic patterns from extensive text corpora, deep learning models—in particular, transformers, convolutional neural networks (CNNs), and recurrent neural networks (RNNs)—have shown impressive performance in natural language understanding (NER) tasks.
The study also addresses current developments in deep learning-based NER models, such as attention mechanisms, transfer learning, and multi-task learning, which have improved the accuracy and resilience of NER systems even more. In addition, it looks at the difficulties and potential paths for NER research, including linguistic knowledge integration into deep learning architectures, managing noisy data, and domain adaptability.
In conclusion, this work opens the door for future research projects and applications in NLP and related domains by offering insights into the state-of-the-art approaches and techniques in Named Entity Recognition[12].
Index Terms—Unstructured Data, Business Analytics, Data Extraction, Natural Language Processing(NLP), Named Entity Recognition(NEP), Deep Learning

1. Introduction

In the field of Natural Language Processing (NLP), Named Entity Recognition (NER) is a cornerstone that is essential to many language processing tasks. Today's data-driven world has made the ability to automatically detect and classify named things inside text crucial. Large amounts of unstructured textual data are scattered over the digital landscape. NER plays a key role in extracting structured information from unstructured text, enabling robots to understand human language more efficiently. This includes information retrieval, sentiment analysis, question answering, and knowledge graph construction.
NER systems were formerly built on statistical models and rule-based techniques, which frequently found it difficult to handle the complexities of natural language and adjust to a variety of textual situations. But the development of deep learning has completely changed the area of NER, bringing with it a new era of scalability and precision never before possible. Deep learning models have become the front-runners in NER research, greatly surpassing traditional techniques thanks to their ability to understand complex patterns and representations from enormous volumes of data.
We also analyze the paradigm change that deep learning in NER has brought about, breaking down the design and workings of neural network models that have redefined the state-of-the-art in entity recognition. We explore the most recent

developments in deep learning-based neural network attention mechanisms, transfer learning, multi-task learning, and other breakthroughs that have pushed the limits of scalability and performance[16].

Additionally, we discuss the prospects and difficulties that the Named Entity Recognition area will face in the future. We examine the directions for further NER research and improvement, from domain adaptation to handling noisy data, from integrating linguistic information to improving model interpretability.

This paper aims to provide a roadmap for researchers, practitioners, and enthusiasts alike, guiding them through the complexities of Named Entity Recognition and fostering innovation in NLP and related disciplines by synthesizing insights from diverse sources and presenting a thorough overview of the landscape.

Named Entity Recognition (NER) finds applications across various domains and industries. Here are some key applications[21]:

- **Information Retrieval and Search Engines**: By allowing users to look for certain things within documents or web pages, NER improves the efficiency of search engines. Search engines can provide more precise and pertinent search results by identifying specified entities, such as individuals, groups, places, and dates.
- **Question Answering Systems:** In order to produce reliable replies, question answering systems require the use of NER. NER is utilized to recognize entities mentioned in questions and obtain pertinent information from text. This program is especially helpful for educational platforms, customer support systems, and virtual assistants.
- **Document summary and Categorization:** By recognizing significant named entities that encapsulate the main ideas or themes in a document, NER helps with document summary and categorization. Large text volumes can now be automatically summarized and categorized, which enhances the organization and retrieval of information.
- **Named Entity Disambiguation:** NER uses context to identify the precise meanings or referents of named entities, hence assisting in their disambiguation. This is essential for clearing up misunderstandings that result from things having different meanings, like "Apple" (the firm) and "apple" (the fruit).
- **Sentiment Analysis and Opinion Mining:** Named entities linked to sentiments or opinions in text are found by NER, which facilitates these processes. This makes it possible for companies to assess public opinion and reputation by examining customer reviews, brand mentions, and social media discussions.
- **Machine Translation and Cross-Lingual Information Retrieval:** By recognizing named items in both source and target languages, NER helps with machine translation and cross-lingual information retrieval. This enhances multilingual communication and worldwide information access by making it easier to translate and retrieve information accurately between languages.
- **Entity Recognition in Biomedical Text:** Biomedical entities, such as genes, proteins, illnesses, and drugs, must be extracted from scientific publications and electronic health records. This is where Neural Entity Recognition (NER) comes into play. Tasks including drug discovery, clinical decision support, and literature curation are supported by this.
- **Legal and Regulatory Compliance:** By locating organizations referenced in agreements, contracts, and regulatory documents, NER assists organizations in maintaining legal and regulatory compliance. This helps in sectors including banking, insurance, and healthcare with contract administration, risk assessment, and compliance monitoring.
- **Social Media Analysis and Trend Detection:** By locating identified entities referenced in social media postings, comments, and reviews, NER assists with social media analysis and trend detection. This makes it possible for companies to keep an eye on mentions of their brand, follow new trends, and interact with clients instantly. In Named Entity Recognition (NER), several technologies are frequently employed, spanning from conventional statistical models to cutting-edge deep learning architectures.

These are some important technologies[12][04]:

- **Rule-based Systems:** To detect named things in text, rule-based NER systems use predetermined language patterns and rules. Patterns for identifying entity kinds, such as names of people or organizations, locations, dates, and more, may be included in these rules.
- **Statistical Models:** Based on statistical patterns seen in annotated training data, statistical NER models detect named things using probabilistic methods and machine learning approaches. In statistical NER, models like Conditional Random Fields (CRFs), Maximum Entropy Markov Models (MEMMs), and Hidden Markov Models (HMMs) are frequently utilized.
- **Conditional Random Fields (CRFs):** NER frequently uses CRFs, a kind of probabilistic graphical model. They are especially useful for capturing sequential dependencies in text, modeling the conditional likelihood of label sequences given input data.
- **Support Vector Machines (SVMs):** NER frequently uses SVMs, which are supervised learning models, for binary classification tasks like identifying named entities from non-entities. Lexical, syntactic, and contextual features are only a few of the characteristics that may be trained on them, and they perform well in managing high-dimensional feature spaces.
- **Deep Learning Architectures:** By making neural network models able to extract intricate patterns and representations from massive text corpora, deep learning has completely changed neural network recognition. NER uses a number of well-liked deep learning architectures, such as:

  - **Recurrent Neural Networks (RNNs):** RNNs have been widely employed in sequence labeling tasks like NER because they are good at capturing sequential dependencies in text, especially Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) variants.
  - **Convolutional Neural Networks (CNNs):** By using convolutional operations, CNNs are good at identifying subtle patterns and characteristics in text. They have been used in NER to extract features from character-level representations and word embeddings.
  - **Transformer-based Architectures:** By utilizing self-attention processes to extract contextual information from huge text corpora, transformer-based models—like the Bidirectional Encoder Representations from Transformers (BERT) and its variants—have attained state-of-the-art performance in NER.

- **Ensemble approaches**: To increase performance overall, ensemble approaches mix several NER models or features. To improve resilience and generalization, ensemble NER systems can benefit from the application of techniques like model averaging, bagging, and boosting, which capitalize on the variety of individual models[7].

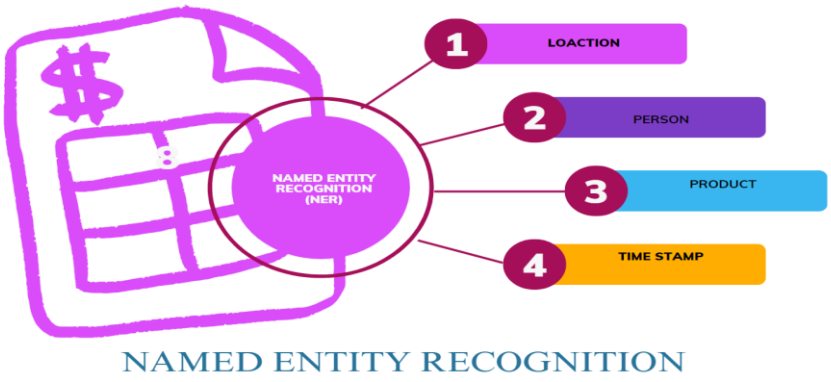

Figure 1: Named Entity Recgnition

2. DATA SETS

A collection of text documents that have been manually tagged with named entities and the associated entity types is referred to as a dataset in the context of Named Entity Recognition (NER). Annotations denoting the boundaries of named entities and their entity types are often appended to each document in the dataset.

In the CoNLL-2003 dataset, for instance, the annotation for a sentence might look like this:

- Sentence: "Mitchel was born in Hawaii."
- Annotations

"Mitchel" - ENTITY: PERSON

"Hawaii" - ENTITY: LOCATION

This annotation shows that "Hawaii" is identified as a location (LOCATION) entity and "Barack Obama" as a person (PERSON) entity[20].

NER models use NER datasets as resources for development, testing, and training. They make it possible for scholars and professionals to train models to identify named entities in text and assess how well they perform in comparison to predetermined criteria. Since high-quality datasets offer annotated examples for model training and assessment, they are essential for creating reliable and accurate NER systems.

2.1. CoNLL-2003:

mong the most widely used benchmark datasets for NER is the CoNLL-2003 dataset. It includes news items from the Reuters corpus labeled with named entity labels for the following four categories of entities: MISC, ORGANIZATION, LOCATION, and PERSON.

The dataset is made up of English news stories from the Reuters corpus that have been named entity labeled for four different entity types: MISC (miscellaneous), PERSON, ORGANIZATION, and LOCATION. The annotations identify the entity types of the listed entities in the text and indicate their bounds. Since the CoNLL-2003 dataset offers a uniform benchmark for comparing the performance of various systems, it is frequently used for NER model evaluation and training. There are three sets of sentences with matching annotations: a training set, a development set, and a test set.

Because it poses a difficult and practical issue in the context of processing unstructured text data, researchers and practitioners in the field of natural language processing frequently use the CoNLL-2003 dataset to create and test NER systems.

2.2. O**neNote:**

An extensive multilingual corpus, the OntoNotes dataset is a useful tool for Named Entity Recognition (NER) and other natural language processing (NLP) applications. It was created as a component of the OntoNotes project, which sought to produce an extensive linguistic resource for numerous languages and genres that was annotated with a variety of linguistic events. Text data from a variety of sources, such as newswire, broadcast news, web text, conversational telephone voice, and more, make up the OntoNotes corpus. It is appropriate for multilingual NLP research since it includes a wide range of languages, including Arabic, Chinese, and English. The OntoNotes dataset offers annotations for named entities in a variety of languages and genres for NER applications. Named entities—such as PERSON, ORGANIZATION, LOCATION, DATE, TIME, MONEY, and more—are annotated with the associated entity types.

A popular dataset in NER research, OntoNotes is used as a standard to assess how well NER systems function in various languages and genres. Its extensive scope, linguistic coverage, and variety of text sources make it an invaluable tool for NER model testing and training in practical settings.

2.3. ACE (Automatic Content Extraction):

In natural language processing, the Automated Content Extraction (ACE) dataset is a popular benchmark dataset, especially for tasks like Named Entity Recognition (NER), entity linking, and connection extraction. It was created as a part of the National Institute of Standards and Technology's (NIST) ACE program, which aims to promote automatic information extraction technologies[14].

The ACE dataset is made up of several kinds of text documents that span a broad range of subjects and domains, such as newswire articles, web data, and broadcast news transcripts. Named entity mentions, entity types, and relationships between entities are annotated in these publications.

The ACE dataset offers annotations for named entities in the context of NER, including GPE (Geopolitical Entity), FAC (Facility), PERSON, ORGANIZATION, LOCATION, and more. Because

every mentioned named entity has its entity type annotated, researchers can use real-world data to train and assess NER models.

The ACE dataset has been extensively utilized in NER research and other information extraction activities due to its superior annotation quality. It functions as a common benchmark for assessing how well NER systems perform, especially in fields where precise entity recognition is essential, including news and information retrieval.

2.4. CONLL++:

The CoNLL++ dataset is an assemblage of datasets that have been aggregated under the CoNLL++ initiative from several sources. It includes datasets utilized in other NLP research projects as well as CoNLL shared tasks. A wide range of languages, domains, and annotation systems are covered by CoNLL++ datasets, which makes them appropriate for a variety of NLP tasks, such as dependency parsing, Named Entity Recognition, and Part-of-Speech (POS) tagging[40].

The domain specificity, quantity, and quality of the annotations in the datasets contained in CoNLL++ can vary. While some datasets provide a wider range of topics and multilingual data, others concentrate on certain fields or languages. Annotated text data with labeled entities, syntactic structures, and linguistic attributes are common elements of CoNLL++ datasets, which enable researchers to create and assess NLP models across various languages and domains.

2.5. ResumeNER:

A specialized collection of resumes, or curriculum vitae, called ResumeNER is annotated with named entities pertinent to the abilities, experiences, qualifications, and other aspects of job candidates. Entities like names of people, organizations, abilities, specifics about schooling, employment history, and contact details are frequently annotated. This dataset facilitates the construction and assessment of Named Entity Recognition (NER) models specifically intended for processing job-related documents. It is specifically optimized for NER tasks in the context of resume parsing and information extraction[11].

3. Tools and Technologies Used for NER

There are numerous Named Entity Recognition (NER) tools and libraries that support various NLP frameworks and computer languages. The following are some popular NER tools:

3.1. spaCy :

Written in Python and Cython, spaCy is an open-source natural language processing (NLP) package. Because of its efficient, quick, and intuitive architecture, it is a favorite among developers for a variety of natural language processing (NLP) tasks, such as dependency parsing, part-of-speech tagging, and Named Entity Recognition (NER). Here are a few of spaCy's salient features:

- Efficiency: SpaCy's memory and speed optimizations make it possible to handle vast amounts of text quickly and effectively. Writing it in Cython allows for high-performance computing because it is a superset of Python that compiles to C.
- Pre-trained Models: For a variety of NLP applications, including NER, spaCy offers pre-trained models. These models can be used right out of the box without requiring a lot of training because they handle various languages and were trained on sizable annotated datasets.
- Personalization: With spaCy, users can train custom models on their own annotated datasets or modify and improve pre-trained models. Because of its adaptability, developers can customize spaCy to fit particular jobs, languages, or domains.
- Tokenization: SpaCy has strong tokenization features, such as the ability to divide text into words, punctuation, and other language components. It is capable of efficiently managing intricate tokenization jobs, including those involving contractions, compound words, and special characters[17].
- Entity Recognition: The NER component of spaCy recognizes named entities in text and places them in pre-established categories like DATE, PERSON, ORGANIZATION, LOCATION, and more. To achieve high entity recognition accuracy, it makes use of statistical models and deep learning architectures.
- Part-of-Speech Tagging: The part-of-speech (POS) tagging feature of spaCy designates grammatical categories, such as noun, verb, adjective, etc., to words in a phrase. Semantic analysis and syntactic parsing are two

downstream NLP activities that benefit from this information.

- Dependency Parsing: spaCy has capabilities for dependency parsing, which examines phrase syntactic structure and expresses word relationships as directed dependencies. This makes tasks like information extraction and semantic role labeling possible.
- Integration: spaCy is simple to add into NLP pipelines and workflows because of its smooth integration with other Python tools and frameworks. Additionally, it offers practical APIs for programmatically processing text input and retrieving linguistic annotations.

3.2. NLTK (Natural Language Toolkit)

A comprehensive open-source package for Python natural language processing (NLP) activities is called NLTK (Natural Language Toolkit). NLTK, which was created by University of Pennsylvania scholars, offers a vast array of resources, tools, and methods for linguistic analysis and text processing. NLTK has the following salient characteristics[36]:

- Text Processing: Tokenization, stemming, lemmatization, and normalization of text data are all possible with NLTK. These tools let developers extract features that are helpful for NLP jobs and preprocess raw text data.
- Part-of-Speech Tagging: Part-of-speech tagging is the process of giving words in a phrase grammatical categories (such as noun, verb, adjective, etc.). NLTK contains pre-trained models and algorithms for this task. Semantic processing, information retrieval, and grammatical analysis can all benefit from this knowledge.
- Named Entity Recognition (NER): NLTK offers resources and tools for named entity recognition, which is the process of locating and categorizing named entities in text data, including people, places, dates, and organizations. In many NLP applications, such as information extraction, question answering, and sentiment analysis, NER is an essential task.
- Syntactic parsing, which examines the grammatical structure of sentences and depicts them as hierarchical structures like dependency networks or parse trees, is supported by NLTK. Semantic analysis, machine translation, and natural language query parsing all depend on this functionality.
- Corpora and Lexical Resources: For a wide range of languages and domains, NLTK offers an extensive collection of lexicons, linguistic resources, and annotated corpora. These resources offer useful data for training and assessing NLP models as well as facilitating NLP research and experimentation.
- Machine Learning: By integrating NLTK with well-known machine learning packages like scikit-learn, programmers may create and hone custom NLP models with methods like sequence labeling, clustering, and classification. Moreover, NLTK offers tools for performance optimization, model assessment, and feature extraction.

3.3. Stanford NER

The Stanford University Natural Language Processing Group created the Java-based software toolset known as Stanford NER (Named Entity Recognizer). Its purpose is to recognize and categorize identified items in text, including people, places, dates, organizations, and more. Stanford NER has the following salient characteristics:

- Pre-Trained Models: For named entity recognition in a variety of languages, including English, Spanish, German, Chinese, and others, Stanford NER offers pre-trained models. These models handle many entity types and languages out of the box and have been trained on extensive annotated datasets.
- Customizable: Stanford NER enables users to train custom models on their own annotated datasets or to modify and improve pre-trained models. Because of its adaptability, Stanford NER can be customized by developers to fit certain domains, languages, or jobs by adding domain-specific characteristics or training sets.
- Efficiency: Stanford NER is renowned for its effectiveness and speed, with the ability to process vast amounts of text in a timely and precise manner. It is appropriate for batch or real-time processing workloads because it makes use of effective algorithms and data structures to maximize memory use and processing performance[]31.

- Stanford NER facilitates the identification of several entity categories, such as PERSON, ORGANIZATION, LOCATION, DATE, TIME, MONEY, PERCENT, and more. It can also be tailored to identify other entity kinds pertinent to particular domains or applications.

  Integration: Named entity recognition technology may be smoothly included into Java applications and workflows by developers thanks to Stanford NER's easy integration. It offers libraries and APIs for text data processing, named entity extraction, and programmatic access to NER models. All things considered, Stanford NER is a strong and adaptable solution for named entity identification in text, including pre-trained models, modification possibilities, effectiveness, and simplicity in integrating with Java apps. It has been widely utilized for many NLP applications requiring precise identification of named things in research, education, and industry.

3.4. AllenNLP

The Allen Institute for Artificial Intelligence (AI2) has created AllenNLP, a state-of-the-art natural language processing (NLP) toolkit built on top of PyTorch. In order to build, train, and implement cutting-edge deep learning models for a variety of natural language processing applications, researchers and developers can utilize this open-source framework, which is modular and adaptable. Its extensive Model Zoo, which houses pre-trained models in a variety of domains like text categorization, named entity recognition (NER), part-of-speech (POS) tagging, and machine reading comprehension (MRC), demonstrates its adaptability. AllenNLP is unique in that it places a strong focus on configurability, enabling users to customize model architectures, hyperparameters, and training configurations using simple configuration files or Python code to meet their unique requirements[33].

In addition to its Model Zoo, AllenNLP is proud of its smooth connection with PyTorch, which gives customers access to the vibrant community of PyTorch for expedited model building and experimentation. Users accustomed to the PyTorch framework will find it easier to transfer because to this synergy with PyTorch, which also guarantees compatibility with current PyTorch codebases and workflows. Additionally, AllenNLP places a high priority on experiment administration, providing resources to track and monitor training progress, record metrics, and use TensorBoard integration to visualize outcomes. These characteristics simplify the iterative process of model building and optimization while also improving reproducibility.

3.5. Flair

Zalando Research created Flair, a sophisticated natural language processing (NLP) platform intended for cutting edge NLP applications. Flair, which is based on Python, provides academics and developers with an easy-to-use platform for training, optimizing, and implementing NLP models. Flair stands out because to its emphasis on contextual string embeddings, which enable more precise and subtle language interpretation by extracting rich contextual information from text input. These embeddings serve as the foundation for Flair's architecture, enabling effective processing of sequential data and producing outstanding results for a range of NLP tasks when paired with sequence tagging models.

Flair offers pre-trained models and components for tasks including text categorization, part-of-speech (POS) tagging, named entity recognition (NER), and more, in addition to its sturdy architecture. These pre-trained models are easily adjustable and adaptable to certain domains or applications by user-specific dataset fine-tuning. Flair's user-friendly interface, copious documentation, and vibrant community support have made it a preferred option for researchers, practitioners, and enthusiasts who want to apply cutting-edge natural language processing (NLP) techniques for their projects and applications.

3.6. Spacy-Custom-NER

Built on top of the popular natural language processing (NLP) toolkit spaCy, Spacy-Custom-NER is a Python library designed especially for custom Named Entity Recognition (NER) operations. With its user-friendly interface and capabilities to expedite data preparation, model training, and evaluation, this library makes the process of training and assessing custom NER models easier. Spacy-Custom-NER lets customers build top-notch NER models tailored to their own domains,

languages, or applications by utilizing spaCy's strong infrastructure[11][12].

Users can simply add named entities to their own text data and train custom NER models on annotated datasets with Spacy-Custom-NER. The library facilitates a range of training approaches, such as building new models from scratch, optimizing pre-trained models, and transfer learning. Additionally, Spacy-Custom-NER facilitates the iterative process of model building and refinement by providing utilities for analyzing model performance, evaluating recall, precision, and F1-score, and visualizing entity recognition results.

In general, Spacy-Custom-NER is a useful tool for practitioners, developers, and academics who want to create NER solutions that are unique to their own requirements. Because of its vast feature set, ease of use, and integration with spaCy, it is a viable option for creating high-quality NER models for a variety of NLP applications.

3.7. Evaluation

Grishman and Sundheim (1996) employed two criteria to assess the performance of NER: type evaluated whether the predicted label was accurate independent of entity borders, and text evaluated whether the projected entity boundaries remained accurate independent of label. The precision, recall, and (micro) F-score for each score category were calculated as follows: precision was defined as the ratio of the system's correctly predicted entities to the total number of entities predicted by the system; recall was defined as the ratio of the system's correctly predicted entities to the total number of entities identified by the human annotators; and the harmonic mean of precision and recall for both type and text.

The exact match metrics consider a prediction to be correct only if the projected label for the full predicted word precisely matches the phrasing of the label (Segura-Bedmar et al., 2013). Relaxed F1 considers the prediction to be accurate if part of the predicted entity is correctly identified. Strict F1 requires that the character offsets in a prediction line up exactly with the input annotations.

The phrases "in Bangalore" and "on August 8," for instance This can be anticipated to be "Bangalore" and "August 8," respectively, which is also acceptable because the system extracts significant data.

False Positive (FP), False Negative (FN), True Positive (TP), and True Negative (TN) can all be used to interpret the metrics mathematically. Precision (represented by Eq. (1)), F1-score (represented by Eq. (3)), and recall (represented by Eq. (2)) are computed using these. These assessment metrics make up the exact-match evaluation, which measures how accurately the model can simultaneously identify its type and boundary[42].

$$Precision = \frac{\mathrm{J}}{\mathrm{J+L}} \quad (1)$$

$$Recall = \frac{\mathrm{J}}{\mathrm{J+M}} \quad (2)$$

$$f1 - score = 2\frac{(\mathrm{Precision})(Recall)}{\mathrm{Precision+Recall}} \quad (3)$$

where K = TN, M = FN, J = TP, and L = FP. An entity that a NER system provides and that is present in the actual data is referred to as a True Positive (TP). An entity that is present in the real data but is not provided by a NER system is referred to as a True Negative (TN). False Positive (FP) refers to an entity that is present in NER systems but not in the actual data. An entity that a NER system does not provide and is absent from the actual data is referred to as a False Negative (FN).

4. NER – Rule Based Approach

Using a rule-based methodology, Named Entity Recognition (NER) identifies and categorizes textual entities using established patterns and linguistic criteria. This approach, which relies heavily on manually created rules and subject expertise, was common in the early phases of NER development. This is a thorough synopsis of NER's rule-based approach:

Components of Rule-Based NER

1. Lexicons and Dictionaries:
- Lexicons are sets of names for individuals, groups, places, etc. that are used to match words in a text.
- Dictionaries: Can be customized for certain domains (e.g., medical terminology, financial firms) and contain common entity names.

2. Pattern Matching:
- Regular expressions are used to identify items

based on patterns, such as phone numbers, email addresses, or dates (e.g., "dd/mm/yyyy").

- Contextual Patterns: Rules that take advantage of the environment in which things exist. For example, titles that come before a person's name, such as "Dr." or "Mr.", can reveal their name.

3. Grammatical Rules:

- Part-of-speech tagging involves defining rules with POS tags that capture items, including capitalized nouns that usually stand for proper names.
- Syntactic Structures: Sentence structure-based rules, such as identifying an organization if the verb "founded" comes after it.

Advantages of Rule-Based NER

- Precision: Excellent precision within narrowly specified domains with distinct patterns and language.
- Control: Gives precise authority over what qualifies as an entity, enabling customization for particular needs.
- Ease of implementation: Generally quite simple to apply to a restricted range of instances.

Difficulties with Rule-Based NER

- Scalability: As the number of rules rises, it becomes more challenging to scale for large and heterogeneous datasets.
- Updating and maintaining this system on a regular basis is necessary as new entities and variations appear.
- Ambiguity: The inability of strict rules to apply in situations involving uncertainty and a variety of linguistic settings.

Example of NER Based on Rules

Think about a rule-based NER system that can identify names and dates in a text. The guidelines may stipulate:

1. Date Identification:

- Regular expression to match dates in forms such as "12/05/2024" or "5-12-24" is \b\d{1,2}[/-]\d{1,2}[/-]\d{2,4}\b.

2. Name Recognition:

- A list of popular first and last names is called a lexicon.
- Pattern Recognition in Context: Identifying names based on titles, such as \b(Mr\.|Mrs\.|Dr\.|Prof\.) [A–Z][a-z]+\b to correspond with "Dr. Jones" or "Mr. Smith".

5. NER using unsupervised learning

Since unsupervised learning does not rely on labeled data, Named Entity Recognition (NER) utilizing unsupervised learning can be quite difficult. To do this, though, a number of methods and strategies have been investigated. The following are some techniques and approaches for carrying out NER without supervision[16]:

1. Clustering-Based Approaches

Similar words can be grouped together using clustering algorithms, which can aid in the identification of named items. Here is a basic synopsis of this methodology:

- Feature extraction: involves taking characteristics out of each text word. Word embeddings, capitalization, word surrounds, and other features are examples of features.
- Clustering: Using the retrieved features as a guide, group related words using clustering methods (k-means, DBSCAN, or hierarchical clustering).
- Entity Extraction: Determine which clusters most likely correspond to named entities by analyzing the data. Named entities could be represented, for instance, by clusters of capitalized words or words that frequently appear in related situations.

2. Distributional Semantics

Make use of distributional semantics, which is predicated on the notion that words that appear in comparable settings typically have comparable meanings.

- Context Window: Compile context words inside a predetermined window size for every word.
- Word Embeddings: To represent each word and its context, use pre-trained word embeddings (such as Word2Vec, GloVe, or FastText).
- Similarity Measures: Determine the scores of similarity between words in order to classify them into entities according to how similar they are in context.

3. Topic Modeling

By identifying subjects in the text, topic modeling approaches like Latent Dirichlet Allocation (LDA) can assist in the identification of entities.

- Text Preprocessing: Tokenize and eliminate stop words from the text.
- Apply LDA to extract topics from the text using topic modeling. Every topic will have a list of words and their corresponding probability.
- Entity Identification: After the topics have been processed, look for terms that are
- probably called entities. Heuristics such as determining capitalized terms or words that commonly occur in particular situations can be used for this.

4. Co-occurrence and Graph-Based Methods

These methods rely on the co-occurrence of words to build a graph and identify entities.

- Co-occurrence Matrix: Construct a co-occurrence matrix of words based on their proximity within a given window in the text.
- Graph Construction: Build a graph where nodes represent words and edges represent co-occurrence relationships.
- Community Detection: Apply community detection algorithms (like Louvain method or Girvan-Newman algorithm) to find clusters of words that co-occur frequently. These clusters might correspond to named entities.

5. Heuristic and Rule-Based Methods

Without labeled data, named items can be successfully identified using straightforward heuristics and rules.

- Capitalization: List all words that don't come at the start of sentences but do start with a capital letter.
- Common Patterns: To identify entities, use regular expressions and common patterns. Dates, email addresses, and particular formats like "Mr. [Name]" or "[Location]" are a few examples.

Work Flow Example

This is a simple sample workflow that combines a few of the methods mentioned above:

1. Text Preprocessing: Perform part-of-speech tagging, eliminate stop words, and tokenize the text.
2. First Entity Extraction: Extract possible entities (e.g., capitalized words) using heuristics.
3. Contextual Similarity: Use word embeddings to represent each word and compute contextual similarities.
4. Clustering: Arrange the terms according to how similar they are in context.
5. Entity Refinement: Using word frequencies and contextual data, refine the clusters to determine which ones are most likely to be named entities.

6. NER using supervised learning

Using supervised learning, Named Entity Recognition (NER) trains a machine learning model to recognize and classify items in a text into specified classes, like names of people, places, dates, and organizations. The first step in this approach is to gather a labeled dataset in which every word in the text has an entity type associated with it. In the statement "John lives in New York," for instance, "New York" would be classified as a location (B-LOC, I-LOC) and "John" as a person (B-PER). Such datasets are commonly created or refined using annotation tools such as SpaCy's annotation tools or Brat's annotation tools[28].

Feature extraction is essential to successfully train the model after the data is ready. The word itself, its part of speech, capitalization, word forms, prefixes, and suffixes are examples of features. In addition, more sophisticated methods use contextual embeddings from transformer-based models such as BERT or word embeddings from models like Word2Vec, which capture the semantic meaning of words in their context. The models that can be chosen include more sophisticated deep learning architectures like BiLSTM-CRF and transformer-based models like BERT or RoBERTa, as well as more conventional machine learning methods like Conditional Random Fields (CRF) and Hidden Markov Models (HMM).

Metrics including precision, recall, and F1-score are used to assess the model's performance after it has been trained on the annotated dataset. Error analysis and hyperparameter tuning are crucial processes for improving the model's performance and lowering misclassification rates. With frameworks such as Flask or Django for API deployment, the finished model can be made available for batch or real-time processing. Periodic retraining with fresh data and ongoing monitoring guarantee that the model retains its

high accuracy over time. Hugging Face Transformers, TensorFlow, PyTorch, Scikit-learn, SpaCy, and Keras are some of the well-liked NER tools and libraries[29].

Below is a simple example using SpaCy to train a custom NER model:

This code provides a basic example to get started with training a custom NER model using SpaCy. For more advanced models and features, consider using frameworks like Hugging Face Transformers.

```
import spacy
from spacy.tokens import DocBin
import random

# Load blank model
nlp = spacy.blank("en")

# Create training data
TRAIN_DATA = [
  ("John lives in New York", {"entities": [(0, 4, "PERSON"), (14, 22, "GPE")]}),
  # Add more data here
]

# Convert training data to DocBin format
db = DocBin()
for text, annot in TRAIN_DATA:
  doc = nlp.make_doc(text)
  ents = []
```

7. NER using deep learning

Deep learning-based Named Entity Recognition (NER) uses sophisticated neural network topologies to automatically recognize and classify items in text. Deep learning techniques, especially those based on Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) networks and, more recently, Transformer models like BERT, are capable of capturing intricate patterns in data and comprehending the context of entities in sentences, in contrast to traditional machine learning techniques. Large volumes of labeled training data, in which textual items are classified as people, places, or organizations, are needed for these models. In the line "Alice visited the Eiffel Tower," for instance, "Eiffel Tower" would be categorized as a place and "Alice" as a person[22].

Tokenization and numeric representation of tokens are two steps in the text preprocessing stage of the deep learning pipeline for NER. Word embeddings from models like Word2Vec and GloVe, or contextual embeddings from models like BERT, which can capture semantic meanings and relationships between words, are frequently used for this. BiLSTM (Bidirectional LSTM) networks are frequently used in common architectures to capture dependencies in both forward and backward directions. On top of this, a Conditional Random Field (CRF) layer is frequently added to predict the best label sequence for a given sentence, guaranteeing that the predicted labels match valid tag sequences.

In order to train these models, parameters must be optimized to minimize loss, a metric that represents the discrepancy between the expected and real labels. GPUs are advantageous for this computationally demanding operation. After training, measures like precision, recall, and F1-score are used to assess the model's performance. Transformer-based models, such as BERT, have

transformed neural network architecture (NER) by offering pretrained models that may be optimized for particular NER applications, leading to cutting-edge performance. These models are suitable for real-time or batch text processing in production contexts. They offer reliable and accurate entity recognition for a range of applications, such as automated customer support and information extraction from massive text corpora[26].

7.1. Text representations

These are representations in which terms from one or more larger corpora are linked to their respective entries in a dictionary by means of their corresponding indices. One-Hot encoding, Count Vectorizer, TF-IDF, Word2vec techniques like skip gram, and the second Continuous Bag of Words (CBOW) are the word representations that are typically utilized.

- For instance, the Automatic Construction of Training Data from social media messaging applications is used in One Hot Encoding in NER. It is quite simple to use and comprehend. Nevertheless, it is unable to rank the words according to importance, which may lead to the creation of a memory- and computationally-intensive sparse matrix.
- The frequency with which a phrase occurs in a document is ascertained by the Count Vectorizer. Condensing a sentence into a single vector, it builds a matrix of reviews and words from the corpus of sentences, populating it with the frequency of each term within each phrase.
- A class of shallow, two-layer neural networks (NN) called Word2Vec has been trained to extract word contexts from language input. For training, pairs of context/target words—where the target word is context—are retrieved from a sizable corpus of words. The phrases that are randomly selected fall inside a certain window encompassing the target word. In this work, we examine the two word2vec methods that were previously mentioned as Skip Grams (Hsieh et al., 2017) and CBOW (Carreras et al., 2003). When dealing with a continuous bag of words, the model is expected to generate the target word using softmax after receiving the context words as input in their embedding form (which is initially initialized randomly).

7.2. Convolutional neural networks

One popular neural network (NN) design that has been thoroughly studied for named entity relationship categorization is the Convolutional Neural Network (Alzubaidi et al., 2021). CNN uses a convolutional operation on the embedding input data to extract the features. Given a sentence = e1, e2, e3,..., en, the embedding input representation is expressed as s = w1, w2, w3,..., wn. CNNs typically come with a number of convolutional filters. Each convolutional filter applies convolution operations to continuous words given by the weight matrix in order to produce a feature based on m-grams[5].

CNNs are utilized for NER tasks as well as image processing jobs. The suggested model by Cho et al. (2020) combines bidirectional Long Short Term Memory for word- and character-level representations with convolutional NN to effectively provide high-level combinatorial feature embedding in biomedical NER. The token format combines character-level features from convolutional neural networks with bidirectional-LSTM to include both local and global attributes at the character level of the embedding. The word-level CNN, character-level, and character-level bidirectional-LSTM representations are then chained and fed into a fully connected (FC) network in order to learn the combination of each representation. The paper also applies an attention strategy to the bidirectional-long short-term memory-CRF model to calculate the similarity between the input tokens, allowing it to foresee the tag of the current token and focus on related tokens inside the phrase. The NCBI Disease and JNLPBA standards datasets were used to further validate the model.

Several NER applications also make use of other CNN variants, such as recurrent neural networks (RNN), gated CNNs, and dilated convolutional neural networks (CNN). For instance, Zhang et al. (2022) combined BiLSTM and dilated convolutional neural network (DCNN) for hierarchical encoding in biomedical NER. They then exploited the advantages of DCNN to effectively record large volumes of data. Multiple feature words are simultaneously inserted into the medical text data in order to improve the performance of medical NER. Extensive experiments conducted on three

real-world datasets demonstrate the superiority of their technique over the comparative models. In a similar vein, Wang et al. (2020) presented the use of ASTRAL, an adversarial trained LSTM-convolutional neural network system, to execute NER algorithms on unstructured data.

In order to obtain text-level representations, CNN is used. Chang and Han (2023) have demonstrated this by using a 3D convolutional neural network for document-level feature extraction rather than a BiLSTM. CNN is able to extract features inside sentences and pays attention to the sequential link among sentences. Because a BiLSTM cannot recognize more than one sentence at a time, it is not possible to get all the information. This means that the use of a CNN is required. In order to optimize each Bi-directional LSTM block of the model, they also investigate a layer-by-layer residual structure, which can solve the deterioration problem that develops as the number of model layers increases.

The Na et al. (2019) model addresses the out-of-vocabulary problem, which typically arises in the neural machine translation (NMT) problem. The OOV (Out of vocabulary) problem was used to examine compositional strategies for Korean NER problems. Using combinational morpheme vectors based on LSTM and ConvNet (Convolutional Neural Network), the work created a novel hybrid representation. To create the hybrid representation of morpheme, the input character vectors are first independently submitted to Long Short Term Memory and ConvNet-based compositions. The resulting compositional morpheme vectors are then concatenated.

For NER purposes, OCR (optical character recognition) data is frequently utilized. A neural network architecture was presented by Zhou et al. (2020b) to extract information from medicine packaging. The data obtained from an OCR is used as a data source by the model. The model is made up of three distinct levels. As a preliminary step in data pre-processing, the language model and seq2seq model comprise the first layer. The sentence that appears in the text is determined by the last layer, which has a CNN in the next layer. The model's evaluation revealed that it had an extra 4%–6% F1-Score.

As previously stated, CNNs were first employed in image processing. Local structure has a crucial role in visuals as adjacent pixels frequently provide significant semantic information. However, words are regularly connected in words but not always in close proximity to one another[16]

8. Recurrent neural networks

Artificial neural networks that are specifically constructed to handle sequential data are known as recurrent neural networks (RNNs). RNNs are especially useful for applications where the sequence and context of inputs are crucial. RNNs are not like standard feedforward neural networks; instead, they feature connections that create directed cycles, which allows them to keep track of information from past inputs in a hidden state. Because of its special design, RNNs can handle data sequences with varying time steps, like time series or natural language sentences, by maintaining context.

The vanishing gradient problem, in which training gradients exponentially decrease over lengthy sequences, poses a serious obstacle to typical RNNs' capacity to learn long-term dependencies. Advanced versions such as Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks were created in response to this. These architectures have components known as gates that control information flow. By preserving pertinent information for longer periods of time, these methods enable them to perform better on tasks requiring long-term memory.

RNNs are widely used in many different fields. They are utilized in natural language processing (NLP), where it is crucial to comprehend the context of words across a sequence for language modeling, machine translation, and sentiment analysis. They are also used in time series prediction, where historical data has a big impact on future results, such predicting stock prices or weather patterns. RNNs are also used in music production by learning patterns in note sequences and in video analysis for tasks like activity recognition.

RNNs are not without limitations, though, especially when it comes to computational efficiency and capturing very long-term connections. With the help of recent

developments like transformer models and attention mechanisms, which let networks concentrate on particular segments of the input sequence and improve parallelization, these problems are starting to be addressed. Due to their notable gains in scalability and performance, these innovations are becoming more and more popular for various sequence-related applications, frequently outperforming conventional RNNs.

9. Transfer Learning

A machine learning technique called transfer learning involves using a model created for one task to serve as the foundation for a model on a different but related activity. This method, which frequently uses less data and computer resources, makes use of the knowledge acquired from the first task to enhance learning efficiency and performance on the subsequent task.

In actuality, transfer learning is commonly used in deep learning, particularly in domains such as natural language processing and computer vision. A model that has been trained on a huge dataset, like as ImageNet, for example, can be optimized for particular applications, such facial recognition or medical picture categorization. The pre-trained model's upper layers, which collect task-specific information, are usually changed or updated to better fit the new task, while the lower layers, which capture generic features like edges and textures, are usually kept[14].

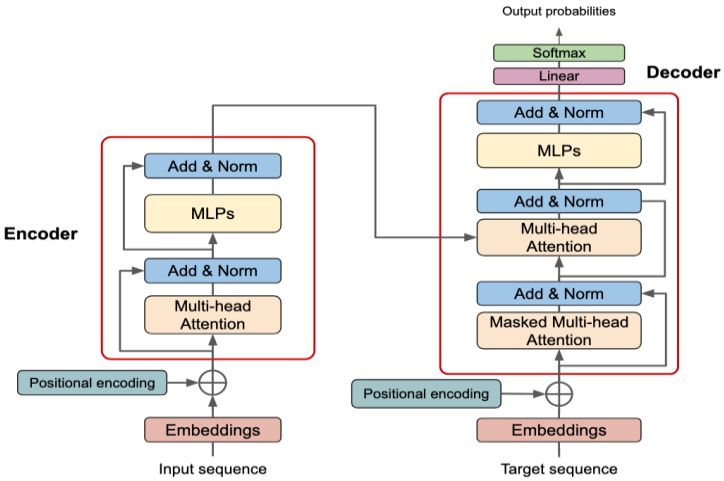

Figure 2: Architecture of a transformer

Transfer learning's primary benefit is that it lessens the requirement for massive volumes of labeled data, which may be expensive and time-consuming to collect. Since the model starts with weights that are already optimized for extracting useful characteristics, it also speeds up the training process. When working with smaller datasets or with constrained computational resources, this is especially helpful.

Because transfer learning makes it possible to apply complex models to specific tasks, it has resulted in major breakthroughs in many domains. It has aided in advances in fields like language translation, where models pre-trained on vast text corpora are refined for particular language pairs, and medical diagnostics, where models trained on generic image datasets are modified to detect disorders from medical scans. Transfer learning contributes to the advancement of machine learning by expanding on previously acquired knowledge.

10. Challenges

One of the most important tasks in natural language processing (NLP) is named entity recognition (NER), which is the process of locating and categorizing entities in a text into predetermined categories. Examples of these categories include names of persons, places, dates, and other proper nouns. Notwithstanding its significance and the progress made in NLP, NER continues to confront a number of formidable obstacles:

1. Ambiguity and Polysemy:

Handling ambiguity and polysemy—the situation in which a single word can have many meanings—is one of the main issues faced by NER. For

instance, depending on the context, the term "Apple" can refer to either the fruit or the technology business. It can be challenging to distinguish between such entities since it takes advanced context knowledge, particularly in texts that are convoluted or badly organized[45].

2. Variability in Entity Representation

There are various methods to refer to entities, such as using acronyms, nicknames, abbreviations, and different forms (e.g., "U.S.A." vs. "United States of America"). Robust normalization techniques and knowledge of the various alternative representations of entities are necessary to distinguish between phrases that relate to the same entity.

3. Context-Dependent Entities

Certain entities can only be accurately identified in relation to their particular setting. For example, depending on the surrounding content, the name "Jordan" could refer to a person, a nation, or even a business. It can be difficult for NER systems to properly use context to discern between these many uses, particularly when there are complex or delicate contexts involved.

4. Lack of Annotated Data

Generating vast amounts of annotated data which can be costly and labor-intensive, is usually necessary for training successful NER models. Annotated data may be scant or nonexistent in many specialized domains, such as legal or medical texts, which makes it challenging to train models that can correctly identify domain-specific items.

5. Handling Multilingual and Code-Mixed Texts

In multilingual settings or when working with code-mixed texts (texts containing multiple languages or language variants), NER becomes more difficult. Since there are differences in the rules and representations of entities between languages, NER systems must be flexible and able to deal with cross-linguistic changes.Code-Mixed Texts and Al

6. Evolving Language and New Entities

Slang phrases, new businesses, and technological advancements are just a few examples of the many new things that constantly change language. For NER systems to reliably identify and categorize these novel items, they must be updated and modified on a regular basis. If static models are not regularly retrained with fresh data, they may soon become antiquated.

11. Future Directions

With the introduction of transformer-based models and deep learning, Named Entity Recognition (NER) has advanced significantly, but there are still a lot of exciting avenues to pursue in the future. Improving contextual awareness is a crucial issue. Even with their proficiency, current models still have trouble understanding ambiguity and complex settings. Subsequent studies could concentrate on more complex models that incorporate wider context windows or make use of more sophisticated attention mechanisms in order to better capture context. Furthermore, investigating multimodal NER may enhance entity recognition and offer richer context by combining data from several sources, such as photos and videos, with text.

Building reliable NER systems for low-resource languages and domains is another exciting avenue. Since English is one of the languages with the most training data available, many NER models may not function well on other languages. Further investigation into few-shot, zero-shot, and transfer learning techniques may aid in the development of more flexible NER systems. These methods could greatly increase the application of NER technology by allowing models trained on one language or topic to be applied to others with effectiveness.

It is imperative that NER tackle the issue of dynamic and developing language in the future. Language is always changing; new words and slang phrases appear on a regular basis. It is recommended that future NER systems integrate methods for ongoing learning and adaptation. This could potentially involve the utilization of online learning approaches, which allow models to be updated gradually with new data as it becomes available. This would guarantee that, despite changes in the language landscape, NER systems stay accurate and up to date[67].

Ultimately, improving NER models' interpretability and explainability is a crucial area for further study. Although deep learning models have demonstrated state-of-the-art performance, it is frequently difficult to explain why a given object was recognized due to the lack of

transparency in their conclusions. It will be essential to develop techniques that offer understandable, concise justifications for NER judgments, especially for applications in delicate or high-stakes fields like the legal and healthcare sectors. This can increase NER systems' credibility and guarantee their wider adoption and dependable use in a range of applications.

12. Conclusion

Named entity recognition is essential for further Information Extraction tasks like ontology population, semantic annotation, and opinion mining, to name a few, in addition to creating scenario templates and relationship identification. We briefly review traditional approaches, current state-of-the-art, and problems. Initially, we gave a quick overview of Named Entity Recognition, including its history, definition, and potential uses. After that, we went over the methods used for Named Entity Recognition. First, let's talk about the Rule-based method and how limited in their generalizability they are. They also need human intervention to be designed, which makes them difficult to apply, and they are limited to particular domains. After that, we go on to additional approaches, like deep learning-based approaches and supervised and unsupervised learning. examining the benefits that various deep learning-based architectures have over the previously stated ones. Feature engineering, less complicated implementation, less need for human interaction, and quicker execution are some of these benefits. Lastly, we assessed the present difficulties and potential paths forward in light of the literature study. This review was conducted to support future research and can be a helpful tool in the development of Named Entity Recognition models based on deep learning.

References

[1] O. P. J. Parthasarathi Pattnayak, “Innovation on Machine Learning in Healthcare Services–An Introduction,” in Machine Learning for Healthcare Applications, 2021. doi: https://doi.org/10.1002/9781119792611.ch1

[2] R. S. A. Usmani, T. R. Pillai, I. A. T. Hashem, N. Z. Jhanjhi, A. Saeed, and A. M. Abdullahi, “A spatial feature engineering algorithm for creating air pollution health datasets,” International Journal of Cognitive Computing in Engineering, vol. 1, pp. 98–107, Jun. 2020, doi: 10.1016/J.IJCCE.2020.11.004.

[3] A. Khamparia, P. K. Singh, P. Rani, D. Samanta, A. Khanna, and B. Bhushan, “An internet of health things-driven deep learning framework for detection and classification of skin cancer using transfer learning,” Transactions on Emerging Telecommunications Technologies, 2020, doi: 10.1002/ett.3963.

[4] S. P. Potharaju and M. Sreedevi, “Distributed feature selection (DFS) strategy for microarray gene expression data to improve the classification performance.,” Clin Epidemiol Glob Health, p. doi:10.1016/j.cegh.2018.04.001, 2018.

[5] A. Ajithkumar and S. Geetha, “Classification of e-commerce financial transaction logs using machine learning approach,” Int J Health Sci (Qassim), pp. 4500–4506, Apr. 2022, doi: 10.53730/ijhs.v6nS1.5835.

[6] H. K. A. M. Sathya , M. Jeyaselvi, Shubham Joshi, Ekta Pandey, Piyush Kumar Pareek, Sajjad ShaukatJamal, Vinay Kumar, “Cancer Categorization Using Genetic Algorithm to Identify Biomarker Genes,” J Healthc Eng, 2022.

[7] O. P. J. Niranjan Panigrahi, Ishan Ayus, “An Expert System-Based Clinical Decision Support System for Hepatitis-B Prediction & Diagnosis,” in Machine Learning for Healthcare Applications, wiley, 2021. doi: https://doi.org/10.1002/9781119792611.ch4 .

[8] U. M. Balakrishnan, K., Dhanalakshmi, R. & Khaire, Hybrid Marine Predator Algorithm with Simulated Annealing for Feature Selection. CRC Press, 2021.

[9] Z. M. H. and D. F. Gillies, “A review of feature selection and feature extraction methods applied on microarray data,” Adv. Bioinf., 2015.

[10] M. Nssibi, G. Manita, and O. Korbaa, “Binary Giza Pyramids Construction For Feature Selection,” Procedia Comput Sci, vol. 192, pp. 676–687, Jan. 2021, doi: 10.1016/J.PROCS.2021.08.070.

[11] P. SankalapAroraa, "Binary butterfly optimization approaches for feature selection," Expert Syst Appl, vol. 116, pp. 147–160, 2019.

[12] U. M. Khaire and R. Dhanalakshmi, "Stability of feature selection algorithm: A review," Journal of King Saud University - Computer and Information Sciences, 2019, doi: 10.1016/j.jksuci.2019.06.012.

[13] R. Nakamura, L. Pereira, K. Costa, D. Rodrigues, J. Papa, and X. Yang, "BBA: A Binary Bat Algorithm for Feature Selection," in 25th SIBGRAPI Conference on Graphics, Patterns and Images, 2012, pp. 291–297, doi: 10.1109/SIBGRAPI.2012.47.

[14] J. T. & A. R. Abdullah, "Binary atom search optimization approaches for feature selection," Conn Sci, p. 10.1080/09540091.2020.1741515, 2020.

[15] L. Abualigah and A. Diabat, "Chaotic binary Group Search Optimizer for feature selection," Expert Syst Appl, vol. 192, p. 116368, Apr. 2022, doi: 10.1016/J.ESWA.2021.116368.

[16] N. Too, J., Abdullah, A. R., & Mohd Saad, "Binary Competitive Swarm Optimizer Approaches for Feature Selection.," Computation, vol. 7, no. 2, p. 10.3390/computation7020031, 2019.

[17] X. Cui, Y. Li, J. Fan, T. Wang, and Y. Zheng, "A Hybrid Improved Dragonfly Algorithm for Feature Selection," IEEE Access, vol. 8, pp. 155619–155629, 2020, doi: 10.1109/ACCESS.2020.3012838.

[18] R. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, "Relief-Based Feature Selection: Introduction and Review," J Biomed Inform, 2017.

[19] N. Neggaz, E. H. Houssein, and K. Hussain, "An efficient henry gas solubility optimization for feature selection," Expert Syst Appl, vol. 152, p. 113364, Aug. 2020, doi: 10.1016/J.ESWA.2020.113364.

[20] S.-C. ; Wang, G.-L.; Chu and J.-S. Tian, A.-Q.; Liu, T.; Pan, "Improved Binary Grasshopper Optimization Algorithm for Feature Selection Problem.," Entropy, vol. 24, p. https://doi.org/10.3390/ e24060777, 2022.

[21] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, "Distributed feature selection: An application to microarray data classification," Appl Soft Comput, vol. 30, pp. 136–150. doi:10.1016/j.asoc.2015.01.035, 2015.

[22] V. Sanchez and N. Bolón-Canedo, "A review of feature selection methods on synthetic data," Knowl. Inform. Syst., pp. 483–519., 2013.

[23] H. Hashemi, A., Bagher Dowlatshahi, M., &Nezamabadi-pour, "A pareto-based ensemble of feature selection algorithms.," Expert Syst Appl, vol. 180, p. doi:10.1016/j.eswa.2021.115130, 2021.

[24] G. Hu, B. Du, X. Wang, and G. Wei, "An enhanced black widow optimization algorithm for feature selection," Knowl Based Syst, vol. 235, p. 107638, Jan. 2022, doi: 10.1016/J.KNOSYS.2021.107638.

[25] and X. Y. B. Xue, M. Zhang, W. N. Browne, "A survey on evolutionary computation approaches to feature selection," IEEE Trans. Evol. Comput., vol. 20, no. 4, pp. 606–626, 2016.

[26] R. J. and Q. Shen, "Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches," vol. 8, pp. 1–19, 2008.

[27] Kashif. Neggaz, Nabil & Houssein, Essam & Hussain, "An efficient Henry Gas Solubility Optimization for Feature Selection.," Expert Syst Appl, p. 152. 113364. 10.1016/j.eswa.2020.113364., 2020.

[28] Q. Gao, Y.; Zhou, Y.; Luo, "An Efficient Binary Equilibrium Optimizer Algorithm for Feature Selection.," IEEE Access, vol. 8, pp. 140936–140963., 2020.

[29] M. Rostami, K. Berahmand, E. Nasiri, and S. Forouzande, "Review of swarm intelligence-based feature selection methods," Engineering Applications of Artificial Intelligence, vol. 100. Elsevier Ltd, p. 104210, Apr. 01, 2021. doi: 10.1016/j.engappai.2021.104210.

[30] Y. Gao, Y. Zhou, and Q. Luo, "An Efficient Binary Equilibrium Optimizer Algorithm for Feature Selection," IEEE Access, vol. 8, pp. 140936–140963, 2020, doi: 10.1109/ACCESS.2020.3013617.

[31] and M. Z. M. C. Lane, B. Xue, I. Liu, "Particle

swarm optimisation and statistical clustering for feature selection," in Advances in Artificial Intelligence (LNCS 8272), pp. 214–220., 2013.

[32] Seyedali. Mafarja, Majdi &Aljarah, Ibrahim & Faris, Hossam &Hammouri, Abdelaziz & Al-Zoubi, Ala &Mirjalili, "Binary Grasshopper Optimisation Algorithm Approaches for Feature Selection Problems.," Expert Syst Appl, vol. 117, p. 10.1016/j.eswa.2018.09.015., 2018.

[33] M. Hambali, T. O. Oladele, and K. S. Adewole, "Microarray cancer feature selection: Review, challenges and research directions," International Journal of Cognitive Computing in Engineering, vol. 1, pp. 78–97, Jun. 2020, doi: 10.1016/j.ijcce.2020.11.001.

[34] T. M. Fahrudin, I. Syarif, and A. R. Barakbah, "Ant colony algorithm for feature selection on microarray datasets," in Proceedings - 2016 International Electronics Symposium, IES 2016, 2017, pp. 351–356. doi: 10.1109/ELECSYM.2016.7861030.

[35] e. a. Gu N, "Efficient sequential feature selection based on adaptive eigenspace model," Neurocomputing, vol. 161, pp. 199–209, 2015.

[36] H. Hichem, M. Elkamel, M. Rafik, M. T. Mesaaoud, and C. Ouahiba, "A new binary grasshopper optimization algorithm for feature selection problem," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 2, pp. 316–328, Feb. 2022, doi: 10.1016/J.JKSUCI.2019.11.007.

[37] J. Tritscher, A. Krause, and A. Hotho, "Feature relevance XAI in anomaly detection: Reviewing approaches and challenges," Front ArtifIntell, vol. 6, p. 1099521, Jun. 2023, doi: 10.3389/frai.2023.1099521.

[38] M. H. Taghian, Shokooh& Nadimi-Shahraki, "Binary Sine Cosine Algorithms for Feature Selection from Medical Data.," Advanced Computing: An International Journal, vol. 10, pp. 1-10. 10.5121/acij.2019.10501., 2019.

[39] M. S. Guha R, Ghosh KK, Bera SK, Sarkar R, "Discrete Equilibrium Optimizer Combined with Simulated Annealing for Feature Selection.," Res Sq, p. DOI: 10.21203/rs.3.rs-28683/v2., 2022.

[40] R. R. L. Venkataramana, S.G. Jacob, "A parallel multilevel feature selection algorithm for improved cancer classification," J. Parallel Distr. Comput., vol. 138, pp. 78–98, 2020.

[41] A. Mitchell, L., Sloan, T. M., Mewissen, M., Ghazal, P., Forster, T., Piotrowski, M., & Trew, "Parallel classification and feature selection in microarray data using SPRINT.," ConcurrComput, vol. 26, no. 4, pp. 854–865. https://doi.org/10.1002/cpe.2928, 2014.

[42] M. Banerjee and S. Chakravarty, "Privacy preserving feature selection for distributed data using virtual dimension," in Proceedings of the 20th ACM international conference on Information and Knowledge Management, ACM, 2011, pp. 2281–2284.

[43] P. K. Ram and P. Kuila, "Feature selection from microarray data : Genetic algorithm based approach," Journal of Information and Optimization Sciences, vol. 40, no. 8, pp. 1599–1610. doi:10.1080/02522667.2019.1703260, 2015.

[44] G. M. and O. Korbaa, "Binary Political Optimizer for Feature Selection Using Gene Expression Data".

[45] L. Jiang, Yugui& Luo, Qifang& Wei, Yuanfei&Abualigah, "An efficient binary Gradient-based optimizer for feature selection," Math Biosci Eng, pp. 3813–3854.

[46] R. H. and H. Kaur, "Binary multi-verse optimization (BMVO) approaches for feature selection," Int. J. Interact. Multimedia Artif. Intell, vol. 6, no. 1, 2020.