SatelliteNER: An Effective Named Entity Recognition Model for the Satellite Domain

Omid Jafari[1][a], Parth Nagarkar[1][b], Bhagwan Thatte[2] and Carl Ingram[3]

[1] *Computer Science Department, New Mexico State University, Las Cruces, NM, U.S.A.*

[2] *Protos Software, Tempe, AZ, U.S.A.*

[3] *Vigilant Technologies, Tempe, AZ, U.S.A.*

Keywords: Natural Language Processing, Named Entity Recognition, Spacy.

Abstract: Named Entity Recognition (NER) is an important task that detects special type of entities in a given text. Existing NER techniques are optimized to find commonly used entities such as person or organization names. They are not specifically designed to find custom entities. In this paper, we present an end-to-end framework, called SatelliteNER, that its objective is to specifically find entities in the Satellite domain. The workflow of our proposed framework can be further generalized to different domains. The design of SatelliteNER includes effective modules for preprocessing, auto-labeling and collection of training data. We present a detailed analysis and show that the performance of SatelliteNER is superior to the state-of-the-art NER techniques for detecting entities in the Satellite domain.

1 INTRODUCTION

Nowadays, large amounts of data is generated daily. Textual data is generated by news articles, social media such as Twitter, Wikipedia, etc. Managing these large data and extracting useful information from them is an important task that can be achieved using Natural Language Processing (NLP). NLP is an artificial intelligence domain dedicated to processing and analyzing human languages. NLP includes many subdomains such as Named Entity Recognition (NER), Entity Linking, Sentiment Analysis, Text Summarization, Topic Modeling, and Speech Processing.

NER focuses on recognizing various entity types such as persons, organizations, products, locations, etc., and these entity types can be different based on the application that NER tool is built for (Nadeau and Sekine, 2007). Some of the early applications of NER included name searching systems in order to identify human names in a given data (Thompson and Dozier, 1997), question answering systems where entities are extracted from the given question in order to find better search results (Florian et al., 2003), and document summarization systems where NER is used to identify important parts of the text (Hassel, 2003). Along with having NER tools for different human languages (e.g., English, French, Chinese, etc.), domain-specific NER software and tools have been created with the consideration that all sciences and applications have a growing need for NLP. For instance, NER is used in the medical field (Abacha and Zweigenbaum, 2011), for the legal documents (Dozier et al., 2010), for tweets (Ritter et al., 2011), and for historical documents (Grover et al., 2008).

Nowadays, many nations around the world are capable of launching satellites into the Earth's orbit. According to a report from (Union of Concerned Scientists, 2020), as of Apr 1, 2020, there are $2,666$ satellites currently in the orbit of Earth. With every satellite launch, tons of new data is generated on social media and news websites. For instance, running the "satellite launch" query in Google News, returns 190 results over only one week, or getting a report for the same query using (talkwalker.com, 2020), tells us that $1.7k$ tweets have been posted in one week. A tool, that can automatically detect satellite entities from different sources of textual data, will be useful for different space agencies and defense institutions around the world. To the best of our knowledge, there is no existing NER tool that can detect satellite entities. The goal of this paper is to improve the performance of detecting satellite entities of state-of-the-art NER tools.

[a] https://orcid.org/0000-0003-3422-2755
[b] https://orcid.org/0000-0001-6284-9251

Jafari, O., Nagarkar, P., Thatte, B. and Ingram, C.
SatelliteNER: An Effective Named Entity Recognition Model for the Satellite Domain.
DOI: 10.5220/0010147401000107
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 3: KMIS*, pages 100-107
ISBN: 978-989-758-474-9

1.1 Contributions of This Paper

Although comparing NER tools has been done before, the existing works (Section 3) have only evaluated the pre-trained and original models of the NER tools, and the evaluation metrics are based on three general entity types. No existing work focuses on improving the accuracy of detecting custom entities (for a given domain) on existing state-of-the-art NER tools. The following are the contributions of this paper:

1. We compare the performance of state-of-the-art open-source and paid NER tools for detecting entities in a specific domain (i.e. satellite domain).
2. We present a workflow for auto-generation of training and testing data for our specific domain.
3. We create a new domain-specific model called *SatelliteNER*, trained using the auto-labeled data.
4. Finally, we present a detailed analysis on the performance of *SatelliteNER* and compare it with state-of-the-art NER tools.

Note that, while this paper focuses on the satellite domain, we believe that the end-to-end methodology for creating an effective custom NER model presented in this paper can be beneficial for other domains as well.

The rest of the paper is organized as follows: In section 2, we give a brief overview of the current methods used in the NER tools. In section 3, we present an overview of the related works to this paper. Section 4 focuses on presenting our end-to-end framework for building our domain-specific NER tool. In section 5, we perform an experimental evaluation of our domain-specific tool against general NER tools, and finally, in section 6, we conclude our work.

2 BACKGROUND

NER approaches can be broadly summarized into two categories: 1. Rule-based approaches, where a grammatical condition is used to detect entities, 2. Machine learning-based approaches where a classifier is built on a large amount of labeled data. Following, we will briefly explain the first two approaches to better get familiar with the concepts used in NER.

2.1 Rule-based

Rule-based approach was first used in (Rau, 1991) where their goal was to identify company names from financial news articles. Rule-based approaches in NER use manual patterns similar to regular expressions in order to match a sequence of words. Instead of matching each word separately using the white space before and after it, it is possible for these methods to identify several words as a single token. There are several phases of rule matchers that will go over the text multiple times and try to identify the entities using partial matches, the context of the words, and full matches. Rule-based methods can achieve high accuracy, but they require significant manual effort by experienced linguists to create hand-crafted rules.

2.2 Machine Learning-based

The main issue with rule-based approaches is that rules are manually handcrafted for a specific domain and not possible to be applied everywhere. Therefore, machine learning-based approaches are being utilized nowadays either in a combination with rule-based methods or by themselves.

Machine learning approaches require a large amount of labeled data to study the features of a given text and generate rules from them. These features provide an abstraction of the text and include data such as word morphology, part-of-speech tags, and local syntaxes. After the features of the given text are found using machine learning techniques, different supervised learning algorithms such as Hidden Markov Models (HMM) (Eddy, 1996), Decision Trees (Quinlan, 1986), Support Vector Machines (SVM) (Hearst et al., 1998), and Conditional Random Fields (CRF) (Lafferty et al., 2001) can be applied to generate matching rules. It is worth mentioning that instead of supervised learning algorithms, semi-supervised and unsupervised algorithms can also be used for this purpose.

One way to provide text features to the supervised learning algorithms is by having an experienced linguistic researcher to generate them, but this process can be automated by using deep neural networks. Two neural network types are mostly used for this purpose: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs were first proposed in (Collobert and Weston, 2008). For RNNs, since they only consider the context right before the input and get biased by the nearest input, an alternative variant called Long Short-term Memory (LSTM) replaced them. LSTMs are capable of learning long-term dependencies, and the bidirectional architecture of LSTMs is capable of also considering the future context. Bidirectional LSTMs were first used in NLP tasks in (Graves et al., 2013) and were later combined with CRF to also use the sentence-level tag information from the CRF layer (Huang et al., 2015).

3 RELATED WORK

In this section, we first introduce the NER tools that are going to be used in our experiments, and after that, we explain the differences of this paper with similar works. The following state-of-the-art techniques are selected for the experiments:

1. Stanford NER (Finkel et al., 2005) was introduced to improve on the NER tasks by using Gibbs sampling and CRF to gather non-local information and Viterbi algorithm for the most likely state inference of the CRF output. The code is implemented in Java and it has the following three models for the English language: 1) "3 class" which supports Location, Person, and Organization entities. It was trained on the CoNLL 2003, MUC 6, and MUC 7 training datasets, 2) "4 class" which supports Location, Person, Organization, and Misc entities. It was trained on the CoNLL 2003 dataset, and 3) "7 class" which supports Location, Person, Organization, Money, Percent, Date, and Time. It was trained on the MUC 6 and MUC 7 datasets. For simplicity and since the entities in the 7 class model are not important in our domain, Stanford NER will refer to the 4 class model for the rest of the paper.

2. Spacy[1] was first introduced in 2015 that used linear models to detect named entities, and it was implemented in Python. Later, as new versions were released, it changed its architecture to neural networks. The latest stable version (v2.3) was released in June 2020 and it uses Bloom embeddings to generate representations of the given text. Furthermore, Spacy uses CNNs since they are computationally cheaper and achieve the same amount of accuracy compared to other neural network architectures. Spacy is designed for production use and has interesting features such as GPU support and transfer learning support. Spacy has three pretrained models for the English language: 1) the small model which was trained on the OntoNotes 5 dataset, 2) the medium model which was trained on a reduced version of the OntoNotes 5 and GloVe Common Crawl datasets, and 3) the large model which was trained on the OntoNotes 5 and GloVe Common Crawl datasets. All of these models are able to recognize PERSON, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK_OF_ART, LAW, LANGUAGE, DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, and CARDINAL entities.

3. Google Natural Language API[2] was first released in 2016 as a general availability version (v1). It is a service provided by Google that is capable of performing several NLP tasks, one of which is called entity analysis that can detect UNKNOWN, PERSON, LOCATION, ORGANIZATION, EVENT, WORK_OF_ART, CONSUMER_GOOD, OTHER, PHONE_NUMBER, ADDRESS, DATE, NUMBER, and PRICE entities. The benefit of using this service is that it lifts the processing power from end-users' machines and all of the processing happens on the Google servers. Therefore, users can easily send API requests via different methods and fetch the results without needing to worry about the processing power of their machine. However, there is a cost associated with this service. The first $5,000$ documents in each month are free and after that, every $1,000$ document costs \$0.50. Google Natural Language also allows users to train their models, but there are separate costs associated with this task and the models will be removed every 6 months.

4. Microsoft Text Analytics API[3] is another cloud-based service which was introduced in 2016. Similar to Google's service, the low-level details of the architecture behind this API is not exposed by the developers. The cost of using this service is free for the first $5,000$ documents each month and after that, every $1,000$ document will cost \$1, and this tool can detect Person, PersonType, Location, Organization, Event, Product, Skill, Address, PhoneNumber, Email, URL, IP, DateTime, and Quantity entities. The downside of service is that up to this date, it does not allow building and training custom models.

5. Stanza (Qi et al., 2020) is a tool developed by the same developers of Stanford NER. Stanza uses a Bi-LSTM neural network trained on the contextualized string representations and a CRF decoder to perform NER. This tool is written in Python and utilizes PyTorch library to enable GPU processing and improve speed. Moreover, its English model is capable of recognizing PERSON, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK_OF_ART, LAW, LANGUAGE, DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, and CARDINAL entities.

There have been several recent papers on the comparison and evaluation of different NER tools. In (Jiang et al., 2016), the authors have selected several NER tools such as Stanford NER, Spacy, etc., and have compared the performance of these tools on a dataset containing annotated PERSON, LOCATION, and ORGANIZATION entities. In (Won et al., 2018), the comparison between NER tools is performed on the historical corpora. Datasets chosen in this study are written in early-modern English and modern English which has helped identify the effect of language

[1]https://spacy.io

[2]https://cloud.google.com/natural-language

[3]https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics

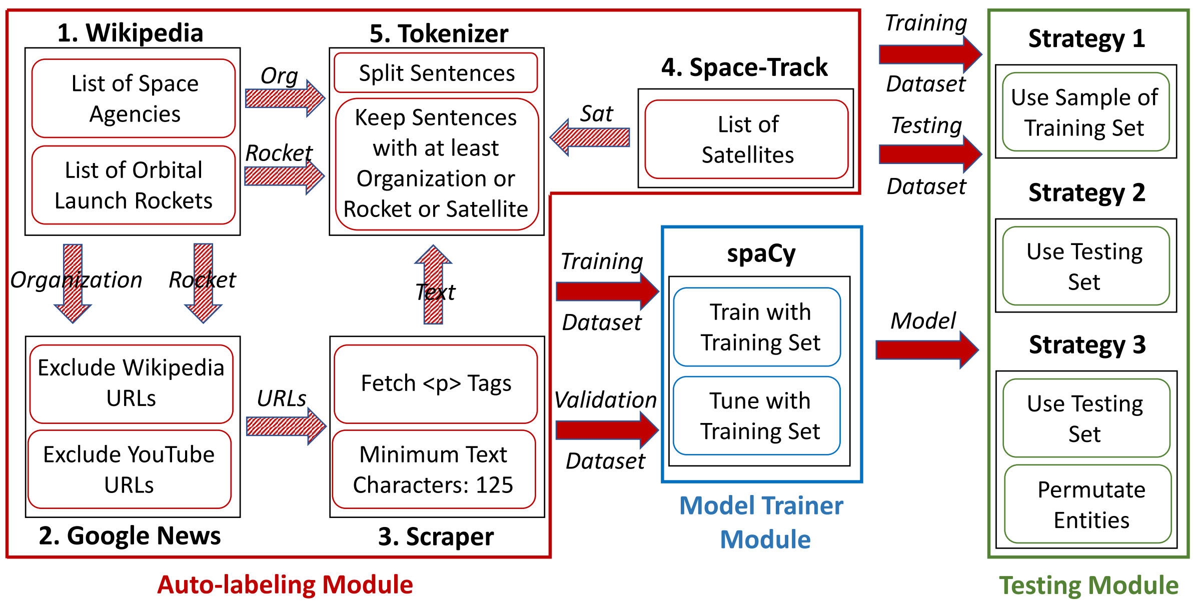

Figure 1: Architecture of SatelliteNER.

changes over time on the NER tools performance. Another work (Schmitt et al., 2019) was published to show the comparison of several NER tools in regards to detecting PERSON, LOCATION, and ORGANIZATION entities in two datasets related to news articles. Finally, in (Ribeiro et al., 2020), authors have proposed a methodology for testing NLP tasks called CheckList. One of the main tests that CheckList introduces is the Invariance test (INV), where changing the entities (e.g. changing person names) should not affect the performance of a tool.

4 SatelliteNER CREATION

In this section, we describe the end-to-end workflow of creating an effective NER model for detecting entities in the satellite domain, called *SatelliteNER*. Figure 1 shows the workflow of *SatelliteNER*.

4.1 Choosing the Base NER Technique

We choose Spacy to build our *SatelliteNER* model since it offers rich features and parameters when training a model. There are two ways to build a domain-specific model in Spacy: 1) Updating the current models and 2) Building from scratch.

The main problem that can happen in updating a current model is known as the Catastrophic Forgetting problem. What happens is that when the model is updated with the new training data, it forgets what was learned before. To deal with this problem, we use the Pseudo-rehearsal strategy, which means that we first use the current model on our training data to find entities and then add those recognized entities to the training data. This way, the model will not forget the weights that were learned before and also its previous entities. As a result, the model will be able to detect the original entities and the ones we add using our training data. However, since there can be common entities in the Pseudo-rehearsal strategy and one of them being chosen (e.g. Product vs SatelliteName), the resultant model in this strategy is not efficient.

For the other strategy, which is building the model from scratch, we must make sure that enough training data is fed into the model. Spacy suggests using at least a few hundred training examples. Therefore, we use an automatic dataset generation workflow which is explained in Section 4.2.

We experimentally show the difference of these two strategies in Table 1. It can be seen that for the reasons mentioned earlier, the second strategy (building the model from scratch) is more effective.

4.2 Efficient Generation of Training and Testing Data

Large amounts of training data is required to build an accurate neural network-based model. For the NER models, the training data should represent the data that we want to predict in the future. For instance, a model trained on news articles is best capable of detecting named entities in news articles. Moreover, the training data should be annotated with the labels that

Table 1: Variations of Trained Models.

Base Model	Psuedo-rehearsal	Training Size	F-Score	Training Time (mins)
None (From Scratch)	No	25,628	91.958	39
Spacy Original Model	Yes	25,628	84.99	142

we want the model to predict in the future. For example, we cannot expect a model to predict organization entities in a text when it was trained only on person labels. All of the NER tools require hundreds and thousands of training data to show a good performance. For example, the OntoNotes 5 dataset that Spacy is using contains $1,445,000$ English words. Manually annotating these large datasets is a very slow process and not applicable to sensitive applications that require ready-to-use systems in a short time. Therefore, we have to come up with an automatic strategy to generate our training data.

After a model is trained on a given data, it can be used to make predictions and detect the named entities. However, the results will be biased if we use the same training data to evaluate the model in the prediction step. To find out that the trained model is capable of not only detecting named entities in the training data but also from any given text (this concept is called generalization in machine learning), we need to prepare an additional dataset called testing dataset. The data in this dataset should be different from our training data, but it should contain the same labels since the model is only capable of detecting those labels. The labels in the testing data are used to evaluate the model and show the performance of it according to different metrics.

All satellites require an orbital launch rocket to send them into the orbit. Also, it is possible for the same model of satellite to belong to several space agencies. For example, Badr satellite belonging to Pakistan's national space agency was launched using a Zenit rocket in 2001 and another Badr satellite belonging to Saudi Arabia was launched on an Ariane rocket in 2008. Our goal is to identify all of these entities; hence, our satellite model is custom built to detect the orgName, rocketName, and satelliteName entities. We use Wikipedia to find governmental and private space agency names.[4,5] Then, we use Wikipedia to find the list of orbital launch systems (i.e. rockets) for each country[6] and choose them for the agencies that we chose in the previous step.

At the next step, we use Google news API to search for every combination in the form of [organization + rocket + "launch"] using the organizations and rockets that we found in the previous step with the goal of finding news articles about the organizations launching those specific rockets. We limit the API to only return the first 250 results and in some cases, the total number of results is less than 250. From the returned results, we remove the URLs containing the "wikipedia", "youtube", ".ppt", or "comments" words because our intention is to only get news articles. So, we end up with $3,243$ URLs without considering the duplicates. Next, we use the BeautifulSoup package to scrape the URLs and getting text contained in <p> tags with a minimum number of 125 characters. The text is also tokenized into sentences using the NLTK package. The result is $129,316$ sentences, but not all of them contain the words that we have searched for; thus, more processing is required.

Since the goal was to also detect satellite names, we use the Space-Track[7] website to get a list of all satellites that have been launched so far. Only the main word within the satellite name is kept (e.g. Starlink-31 and Starlink-32 both become Starlink) and we carefully review the names and remove the common names such as "launch", "step", "wind", etc.

At this step, we analyze the scraped sentences and using the list of organizations, rockets, and satellites, we only keep sentences containing at least one of those (since we have removed numbers from satellite names, we check 95% similarity instead of an exact match for satellite names). As a result, we get $26,128$ sentences out of which $25,628$ of them are used as our training data. Moreover, we also need evaluation data in the training phase that will be used to check the model accuracy and losses at each iteration and tune hyper-parameters. We use the remaining 500 sentences of the results for this purpose.

For the testing dataset, we use three strategies: 1) We use 500 of the sentences from the training set (these are the sentences that model has seen before in the training phase), 2) We use the auto-labeling technique that we used to generate our training set with new organizations and rockets (since the organizations, rockets, and URLs will be different from the training set, they will also contain different satellites and these data will be new to the model) and randomly choose 500 sentences from the results, and 3)

[4]https://en.wikipedia.org/wiki/List_of_government_space_agencies

[5]https://en.wikipedia.org/wiki/List_of_private_spaceflight_companies

[6]https://en.wikipedia.org/wiki/List_of_orbital_launch_systems

[7]https://www.space-track.org

The third strategy is similar to the evaluation done in (Ribeiro et al., 2020) which is called invariance testing. In this strategy, we use the same sentences in the second strategy but randomly shuffle the entities. Since we have three entities and in some sentences, only one of them exists, we first create a list of all entities in the training set. Then, for each sentence, we randomly replace the entities with the entities from the created list and remove them from the list. This way, we can make sure that all entities in the sentences are changed.

5 EXPERIMENTS

In this section, we evaluate the models that we trained and compare them with state-of-the-art NER tools. Experiments were run on a machine with the following specifications: Intel Core i7-6700, 16GB RAM, 2TB HDD, Python 3.6, and Ubuntu 18.04 operating system. All experiments are executed on one core of CPU except Stanza which uses all cores and does not have an option to change the number of cores. Also, default parameters are used for all experiments. We compare our trained model with the following alternatives:

- **Spacy_Orig:** This alternative is the original English model of Spacy.
- **StanfordNer:** As stated in Section 3, the 4 class model of Stanford NER in used our experiments.
- **Stanza:** We use English model of Stanza.
- **GoogleNER:** Here, we use the Cloud Natural Language API service from Google.
- **MicrosoftNER:** We use the Text Analytics API service from Microsoft.

5.1 Evaluation Criteria

Since the tools have different entity labels (e.g. rocketName in our model vs PRODUCT in the original model of Spacy), it is not possible to compare them and perform the evaluation based on them. For instance, both rockets and satellites in our model, can show up as a PRODUCT in the other tools. As a result, we use the following metrics in our evaluation:

- **Precision:** In our testing data, we have the organization name, rocket name, and satellite name labels, and as long as those entities are returned by the different tools under any entity type, we count them as a true positive. Moreover, it is highly possible that tools and models return entities that are neither related to our domain nor equal to the testing data. We define these entities as false positives. The precision metric will show the ratio of true positives over the total number of detected entities. In other words, this metric will help us understand how well a model is in returning only the true positives and not unrelated entities.
- **Recall:** The entities in the testing set that should be detected by the NER tools but not detected by them are called false negatives. The recall metric will tell us how well a model is in finding all of the true positives and is a ratio of true positives over all entities in the testing set.
- **F1 Score:** There are times that improving on recall can result in a decrease in precision. The F1 Score (also known as F-measure) will combine precision and recall and find a weighted average of them. This metric can be used to show the overall performance of a tool.
- **Processing Time:** The time taken to recognize the entities is also an important factor especially in real-time applications. Some tools require loading different models and some require making API calls. We consider all of the aforementioned times as the processing time.

5.2 Discussion of the Results

Precision: Figure 2 shows the precision for all of the techniques. *SatelliteNER* has the best precision since it is only detecting the entities that we are looking for, and thus, has a very low number of false positives. All other NER tools are behaving similarly in terms of precision and return similar number of false positives.
Recall: Figure 3 shows the recall of the NER techniques. We observe that *SatelliteNER* has a high recall in testing dataset 1 since this is the dataset it was trained on and a recall of more than 50% in testing datasets 2 and 3. Stanza and Spacy_Orig have a better recall compared to StanfordNer since they utilize neural network-based models. The architecture and neural network of MicrosoftNER and GoogleNER are not released but results show that GoogleNER is performing better in these testing datasets.
F1 Score: Figure 4 shows the F1 Score results. Since F1 Score is a weighted average of Precision and Recall and due to its high precision, *SatelliteNER* has the highest F1 Score in the tested datasets. Furthermore, all other tools have a similar F1 Score with Stanza and Spacy_Orig being slightly better.
Processing Time: Figure 5 shows the total processing time required to detect entities in a given text. We feed the sentences one at a time to the NER tools and

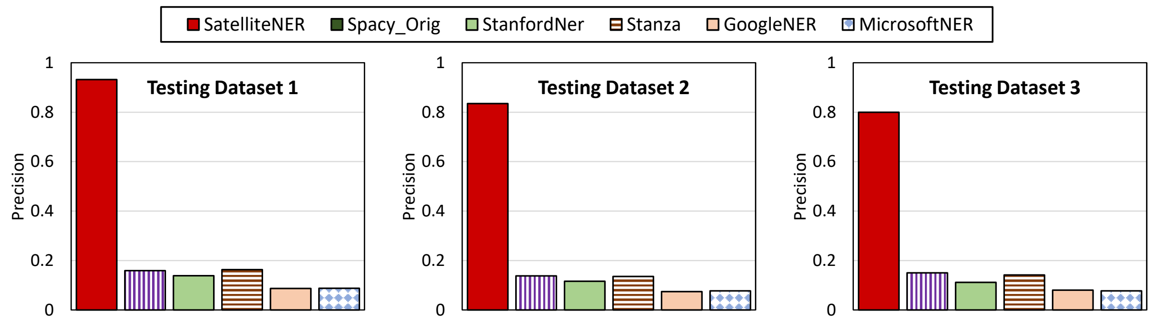

Figure 2: Comparison of the Precision of the SatelliteNER model against alternative tools.

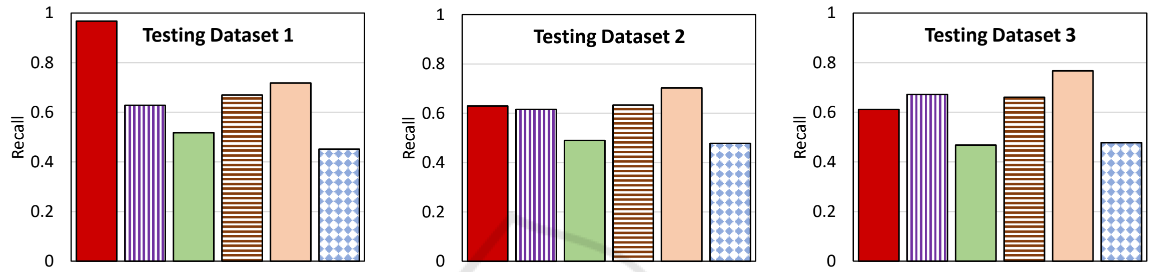

Figure 3: Comparison of the Recall of the SatelliteNER model against alternative tools.

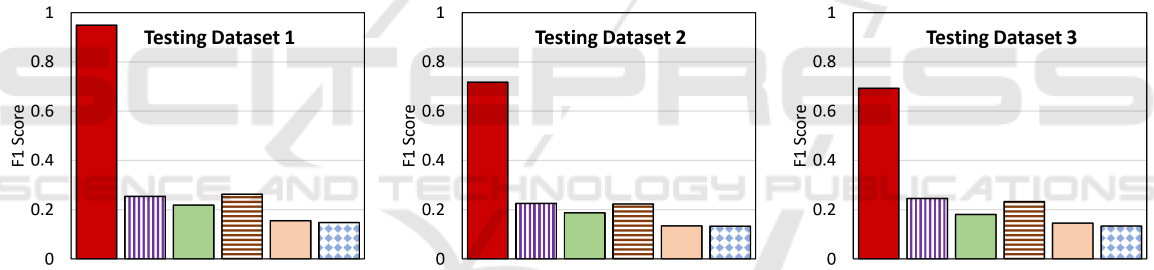

Figure 4: Comparison of the F1 Score of the SatelliteNER model against alternative tools.

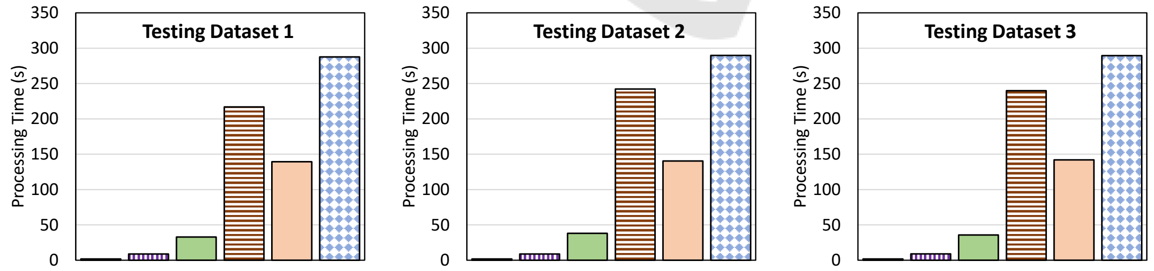

Figure 5: Comparison of the Processing Time of the SatelliteNER model against alternative tools.

evaluate the returned entities. While Figures 2, 3, and 4 show that Stanza has a better accuracy, it is significantly slower than Spacy_Orig since the underlying neural network in Stanza (BiLSTM) is slower than the network in Spacy_Orig (CNN). Also, Stanza mentions in its documentation that it performs very slow when sentences are fed to it one by one instead of a whole corpus; however, for the experiments to be fair, we use the same testing methods for all NER techniques. MicrosoftNER and GoogleNER are slow since they are API-based tools and require network operations to send the requests and receive the results. Moreover, since *SatelliteNER* has to only look for three entities (i.e. Organizations, Rockets, and Satellites), it is faster than Spacy_Orig.

6 CONCLUSIONS AND FUTURE WORK

Named Entity Recognition (NER) is a famous task in Natural Language Processing that is aimed at detecting different entities in a given text. Natural Language Processing analysis in the satellite domain is necessary because of the increasing data growth in this domain and also the importance of this domain. In this paper, we present an effective NER model specifically engineered for the satellite domain called *SatelliteNER*. To build this model, we generate training, validation, and testing datasets in an automated manner. By doing this, the dataset annotation can happen at a fast pace without the need for a human to manually perform the annotation task. Experiments using three different testing strategies show the benefit of *SatelliteNER* over existing NER tools. In the future, we plan to improve accuracy by including human-in-the-loop in the labeling process and by fine-tuning the underlying neural network parameters. Furthermore, we intend to build transformer-based custom models that can achieve a higher accuracy.

REFERENCES

Abacha, A. B. and Zweigenbaum, P. (2011). Medical entity recognition: A comparaison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, pages 56–64.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Dozier, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., and Wudali, R. (2010). Named entity recognition and resolution in legal text. In *Semantic Processing of Legal Texts*, pages 27–43. Springer.

Eddy, S. R. (1996). Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.

Finkel, J. R., Grenager, T., and Manning, C. D. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 168–171.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Grover, C., Givon, S., Tobin, R., and Ball, J. (2008). Named entity recognition for digitised historical texts. In *LREC*.

Hassel, M. (2003). Exploitation of named entities in automatic text summarization for swedish. In *NODALIDA'03–14th Nordic Conferenceon Computational Linguistics, Reykjavik, Iceland, May 30–31 2003*, page 9.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jiang, R., Banchs, R. E., and Li, H. (2016). Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*, pages 21–27.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rau, L. F. (1991). Extracting company names from text. In *Proceedings The Seventh IEEE Conference on Artificial Intelligence Application*, pages 29–30. IEEE Computer Society.

Ribeiro, M. T., Wu, T., Guestrin, C., and Singh, S. (2020). Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1524–1534.

Schmitt, X., Kubler, S., Robert, J., Papadakis, M., and LeTraon, Y. (2019). 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.

talkwalker.com (2020). https://www.talkwalker.com/social-media-analytics-search.

Thompson, P. and Dozier, C. (1997). Name searching and information retrieval. In *Second Conference on Empirical Methods in Natural Language Processing*.

Union of Concerned Scientists (2020). https://www.ucsusa.org/resources/satellite-database.

Won, M., Murrieta-Flores, P., and Martins, B. (2018). ensemble named entity recognition (ner): evaluating ner tools in the identification of place names in historical corpora. *Frontiers in Digital Humanities*, 5:2.