Named Entity Popularity Determination using Ensemble Learning

Vikram Karthikeyan
Department of ISE
BMS College of Engineering
Bangalore, India
vikram.is17@bmsce.ac.in

Ranjan Samal
Department of Multimodal NLP
Samsung R&D Institute
Bangalore, India
ranjan.samal@samsung.com

B Shrikara Varna
Department of ISE
BMS College of Engineering
Bangalore, India
bshrikara.is18@bmsce.ac.in

Shambhavi B R
Department of ISE
BMS College of Engineering
Bangalore, India
shambhavibr.ise@bmsce.ac.in

Amogha Hegde
DDepartment of ISE
BMS College of Engineering
Bangalore, India
amoghahegde.is17@bmsce.ac.in

Jayarekha P
Department of ISE
BMS College of Engineering
Bangalore, India
jayarekha.ise@bmsce.ac.in

Govind Satwani
Department of ISE
BMS College of Engineering
Bangalore, India
govind.is18@bmsce.ac.in

Abstract

Determining the popularity of a Named Entity after completion of Named Entity Recognition (NER) task finds many applications. The most popular of them being virtual assistants where disambiguating entities without contextual help is crucial. A single named entity could belong to multiple domains, making it necessary for the popularity determination approach to give accurate results. The more accurate results can be used to improve the functioning of virtual assistants. This work studies disambiguation of Named Entities (NE) of Music and Movie domains and resolves popularity considering relevant features like region, movie/music awards, album count, run time etc. Decision Trees and Random Forests approaches are applied on the dataset and the latter ensemble learning algorithm resulted in acceptable accuracy.

1 Introduction

Over the years the use of electronic devices for various tasks and services have become predominant. These days, services such as E-commerce, video and music streaming over the internet are common. Thus, it is crucial that these services provide the users with what they require and also give relevant recommendations. Moreover, search engines should know what results to show based not only on the query string but also consider other factors like demographic, geographic location, user's search history, etc. Identifying entities in a query string is termed as Named Entity Recognition (NER). Due to the vast and diverse collection of data, NER sometimes alone may not be sufficient. In such cases Name Entity Popularity Detection can be used to prioritize results which may be more rel-

Proceedings of 17th International Conference on Natural Language Processing: Workshop, pages 27–32
Patna, India, December 18 - 21, 2020.

evant to the user based on various factors. This is especially used while designing a conversation smart assistant, to provide the user with the best results. For example, if the user asks the virtual assistant to "play wolves" then it has to decide whether to play the song wolves or play the movie wolves without any given context or sentence associated with. In such a scenario information about the user's search history has conventionally been used to resolve the ambiguity and judge what the user really wanted. This process of using data other than the query string to disambiguate between entities with similar names is known as Named Entity Popularity Determination (NEPD). Our objective is to accomplish this task without considering user history.

2 Problem Statement

NEPD aims at determining the popularity of the Named Entities. This approach can be extended for use in Conversation Smart Assistants, helping the application to understand user speech, disambiguate entities and give the best search results. NEPD involves primarily

- Data Mining of features which help to determine the popularity of the target NE.
- Designing an algorithm to predict the popularity from the mined features for various domains. Music and Movie domains are considered in our work.

3 Related Work

Several approaches have been put forward to get better results for NER. Some of these methods involve the use of neural architectures in addition to Bi-LSTM methods. Lample et al. (2016) has developed a method based on transition-based parsing and stack-LSTMs. Building on the approach of ensemble learning for NER, multiple approaches have been combined to obtain a better result (Speck and Ngomo, 2014). The results of various classifier algorithms were integrated (Florian et al., 2003), resulting in significant strides in NER performance.

In the work of (Cucerzan, 2007), information extracted from Wikipedia is stored in two databases, and the entities are mapped together. After entity mappings, a disambiguation component is used for Named Entity Disambiguation (NED). To improve the performance of NED, in addition to Bi-LSTM; other models that use GCN and RNN, along with

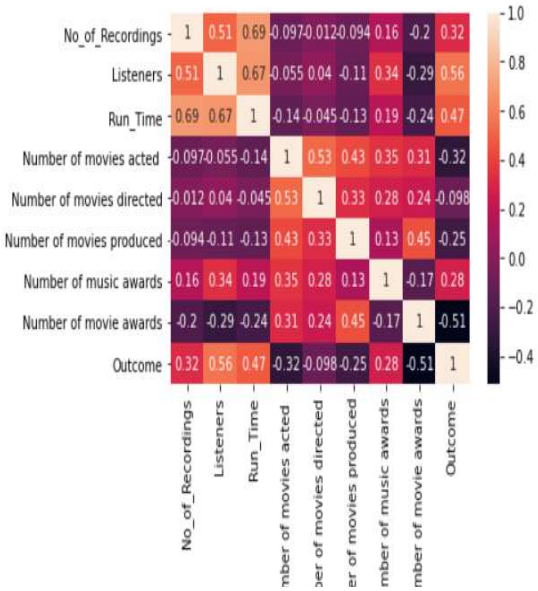

Figure 1: Correlation Matrix of the features

attention have also been experimented upon (Cetoli et al., 2018). Combining graphs and popularity ranking into a single model is another approach towards NED (Alhelbawy and Gaizauskas, 2014) , (Han and Zhao, 2010), where graphs, knowledge bases, are used for Entity Generation, Entity ranking and NED. Joint Embeddings have also been used for improving the results for NED (Yamada et al., 2016).

To enhance the results of the NER models, Entity Popularity models have been proposed (Govani et al., 2013), (Blanco et al., 2013). These models use the personal history of the user, create a ranking of entities, and have resulted in recommendations that are more likely and specifically related to the concerned user. The ranking of entities has been improved by increasing the performance of the Language Model (LM), during Automatic Speech Recognition (Van Gysel et al., 2020). The model used for predicting entity popularity in this paper resulted in improvements of 20% in word error rate.

4 Dataset and Feature Extraction

The next step in dealing with the sparseness was building the correlation and covariation matrix.The correlation matrix, shown in *Figure1* helped decide which features to drop, and which features to retain, to get a better clarity of the data we had collected.

The initial dataset of entities and the domains to which they are to be classified into movie/music

Feature Name	Description	Souce
Type	Genre (applicable to musicians,songs and movies only, and not to actress	IMDb, MusicBrainz
Count of Movies Acted	The number of movies a person has acted in (i a significant role)	IMDb
Count of Movies	Directed Some artists tend to direct the movies they work in.	IMDb
tCount of Movies Produced	Some artists tend to produce the movies they work in. Also includes the number of albums that are self produced by an artist.	IMDb
Release Date	The time description of when a particular song or movie was released. It helps in determining the popularity of that particular work in that decade.	IMDb, MusicBrainz and Last.fm
Region	Place of release of the movie or album or location of concert also refers to the region where an artist is based.	IMDb
Count of Albums	The higher the number, the more weightage for music domain	Last.fm API
cCount of Concerts	The number of live concerts held by the particular artist	Last.fm API
Count of Movie Awards	The number of awards a particular person has in a domain	IMDb
Count of Music Awards	The number of awards a particular person has in a domain	IMDb, MusicBrainz
Run Time	Sum of durations of all the songs released by the person	MusicBrainz

Table 1: Features and their description.

was generated. Considering the domains in the problem statement, a list of applicable features along with the source from which the data for that particular feature can/could be extracted was made. This decision was based on the considerations that a person who is active predominantly in one of the two domains, will have a higher value for the features pertaining to that domain. Additionally, people who are active in both the domains will have values for all the features but some comparatively higher than the other. The list of features is given in *Table 1*.

The problem at hand is finding the popularity of a person in the domain and identification of the domain in which the person is more popular if the person has an existence in both. The idea behind the selection of a feature was based on this problem. The popularity of any person in a domain is based on the works of that person, and the accolades the person has received for their work in the particular field. The feature "Run Time" was included in particular as some of the artists voice for songs in the movies they have acted in. And hence they might have a considerable number of songs under them. This feature will help us to distinguish between a full-fledged singer (music domain) and actors who just give voice for a song in a movie.

The data was collected and a .csv file was formed. Most of the entities in the given dataset were disambiguous, and hence the data collected was sparse as the entity was inclined to one of the domains. To overcome this problem of sparseness, the dataset was first divided into two-parts: one file containing the entities that were the names of the people and/or music bands, while the other contained the entities that were the names of the artwork (movie and/or song tracks). Dividing the single file into two, helped deal with the sparseness a bit as some of the features that were not applicable to that particular category were removed.

There were many challenges in handling the artwork file. Many artworks pertaining to the same

domain had the same name but different artists. A disambiguation had to be done to select the best one of these. Consider the song "Bad Guy". When we tried to collect data regarding it, the result of the search query included 280 songs of the same name in all languages combined. We took the collection of top 10 searches and the data related to each of those songs which shared the common name.

The features "Genre", "Region" and "Release date" were dropped from consideration. The decision to drop them was taken based on the fact that "Genre" didn't work well with entities belonging to the movie domain. The "Release Date" isn't applicable to persons but only to the artwork. Hence these features were considered as not needed and were dropped.

After dropping the not required features from the dataset, the sparseness that still existed was dealt by filling them up with default values which was determined in such a way so as to not change the dynamics of the dataset (i.e., not change the nature of the data).

5 Methodology

Flowchart in *Figure2* shows the systematic approach taken. The data collected from IMDb and MusicBrianz was analyzed. We faced the problem of excessive sparseness after building our dataset as some fields were not applicable for a particular domain and hence had to be left blank. Features like Number of movies acted, Number of movies directed, Number of movies produced would not be applicable for music related entities. Similarly features like release region, release year would not be applicable for movie artists.

We realized adding mean/median values to remove the sparseness would corrupt the dataset as the missing values were not applicable for the entity. For entities like number of movies acted, number of movies directed, number of movies produced we replaced Null values with zero. For the release year entity, we replaced Null values with 2015 and release region with USA to avoid ambiguity as we didn't want these values to influence the classifying decision. We replaced the Null values in the rating columns for movie artists to four to maintain uniformity. We made sure not to replace any missing attribute's value with an extreme value so as to avoid influencing the classifying decision.

We used Label encoding to encode the attribute values as decision trees and random forests do not

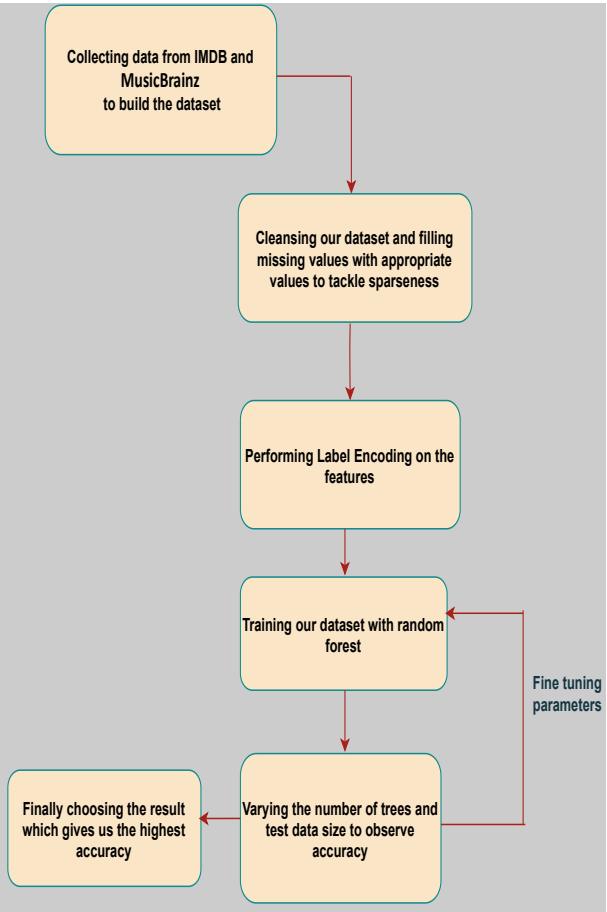

Figure 2: Flowchart of the System Processes

accept string inputs. We used Label Encoding over One Hot encoding because we observed One hot encoding would result in higher data duplication (Multiple Columns). Initially we used decision trees after performing label encoding on the dataset entities. The intuition behind choosing decision trees was that the algorithm generates rules for classification. The input data was not sequential for us to try Machine Learning algorithms like RNN or LSTM. The decision tree was built using the scikit-learn library of python. The algorithm used to build the decision tree was CART, and hence categorical values weren't supported and encoding had to be done. The decision of splitting the node at a level was based on the gini impurity of the features.

To improve the accuracy, we used random forest (Set of decision trees) to reduce overfitting and give better results for new test data. Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better

than a single decision tree because it reduces the over-fitting by averaging the result and improves the classification accuracy.

6 Experimental Results

On analyzing our dataset, we decided that a machine learning approach would work well for the problem in hand. We initially used decision trees and achieved an accuracy of 86%. *Figure3* represents the decision tree that was built using the CART algorithm. The split of nodes at each level is depicted along with its Gini impurity value.

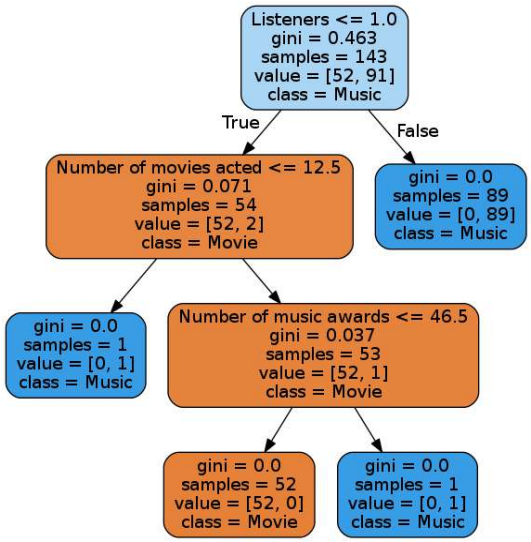

Figure 3: Decision Tree generated

Algorithm	Test Data (%)	Training Data (%)	Accuracy (%)
Decision Tree	20	80	86
Random Forest	20	80	89
Random Forest	30	70	93
Random Forest	40	60	94.3

Table 2: Experimental Results.

To increase the accuracy of classification and prevent overfitting we decided to use random forest. Using the random forest, we obtained an accuracy of 89%. As our dataset was relatively small, increasing the number of trees for random forest did not yield us with better accuracy, so we increased the test data size to have larger data to classify from 20% to 40%. Results are tabulated in Table 2. Finally, we achieved an accuracy of 94.3% percent using random forest with 100 trees.

7 Conclusion

The problem of Named Entity Popularity Determination was to be solved without any contextual data or user history in the domains of music and movies. In virtual assistants, users usually give only named entity without context or a statement associated with it. We built our customized dataset from IMDb and MusicBrainz. The issue of excessive sparseness was solved by filling the missing values in the dataset with generic relevant values and made sure it wouldn't influence the classifying decision. With decision trees, an accuracy of 86% was achieved. To improve the accuracy and prevent overfitting random forests was used. We finally achieved an accuracy of 94.3%. The approach used here to disambiguate ambiguous entities can be extended to other related domains like TV Shows, Podcasts and Radios by collecting relevant features. This approach can be modelled to disambiguate ambiguous entities between various related domains without contextual information.

References

Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 75–80.

Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. 2013. Entity recommendations in web search. In *International Semantic Web Conference*, pages 33–48. Springer.

Alberto Cetoli, Mohammad Akbari, Stefano Bragaglia, Andrew D O'Harney, and Marc Sloan. 2018. Named entity disambiguation using deep learning on graphs. *arXiv preprint arXiv:1810.09164*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 708–716.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 168–171.

Tabreez Govani, Hugh Williams, Jamie Buckley, Nitin Agrawal, Andy Lam, and Kenneth A Moss. 2013. Determining entity popularity using search queries. US Patent 8,402,031.

Xianpei Han and Jun Zhao. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 50–59.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

René Speck and Axel-Cyrille Ngonga Ngomo. 2014. Ensemble learning for named entity recognition. In *International semantic web conference*, pages 519–534. Springer.

Christophe Van Gysel, Manos Tsagkias, Ernest Pusateri, and Ilya Oparin. 2020. Predicting entity popularity to improve spoken entity recognition by virtual assistants. *arXiv preprint arXiv:2005.12816*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.