Can neural decision trees applied to natural language processing improve the parsing and execution of command recognition models for accessibility?

Abstract:
Command-recognition models have long played a critical role in accessibility-focused applications, where accurate and transparent interpretation of user commands is vital. This paper examines whether neural decision trees, a hybrid approach combining the interpretability of traditional decision trees with the representational power of deep neural networks, can meaningfully improve the parsing and execution of commands within natural language processing (NLP) tasks. We trace the field's evolution from early rule-based and statistical models to modern deep-learning and ensemble techniques. Building upon theoretical insights, we conduct an empirical study using a synthetically generated dataset that simulates real-world linguistic variability and noise, then vectorize inputs via TF-IDF and compare the performance of four models—Multinomial Naive Bayes, Decision Tree, Random Forest, and Neural Decision Tree—across key metrics. Results show that ensemble models, particularly the Neural Decision Tree, outperform single models, offering high accuracy, precision, recall, F1-score, and ROC-AUC. These findings underscore the strengths of neural decision trees in capturing complex linguistic structures and demonstrate their value for accessibility-based command recognition. In conclusion, the paper discusses how further work can incorporate more diverse datasets, multimodal inputs, and advanced interpretability features to build even more robust, inclusive, and trustworthy NLP systems.

Literature Review

Over the last twenty years, progress in natural language processing (NLP) has greatly improved the creation of command recognition models, especially for accessibility purposes. As people want systems to be more accurate, efficient, and easy to understand, neural decision trees have become a hopeful solution. They connect old decision trees with new deep-learning methods. This review looks at how command recognition models have changed, focusing on using neural decision trees and mixed NLP techniques to better understand and carry out commands. It examines basic studies, important breakthroughs, and top models while showing how they work for accessibility.

Early Approaches: Rule-Based Systems and Statistical Models

At first, command recognition research used rule-based systems and statistical methods. Katz (1987) introduced n-gram language models to predict word sequences. These models were good for simple commands but struggled with unclear meanings and lacked awareness of context, which limited their use in real situations. Accuracy was usually between 50-60% for complex or multi-part commands.

A big step forward happened with the Hidden Markov Model (HMM) for speech and command understanding, as shown by Rabiner (1989). HMMs gave a way to better parse commands by using probabilities, raising parsing accuracy to about 70% by considering time dependencies. Still, these systems needed a lot of manual setup and often didn't work well in noisy or challenging environments for those needing easier access.

Transition to Machine Learning Models

The move from rule-based systems to machine learning was a key point in command recognition research. Support Vector Machines (SVMs) and decision trees started to be used for command tasks. Quinlan (1996) introduced the C4.5 algorithm, which helped create decision tree-based systems. These models were good at showing clear decision processes but couldn't handle big, complex NLP data.

For accessibility-focused uses, researchers like Begum et al. (2004) studied decision trees to understand voice commands from users with speech problems. These systems were easy to understand and use but depended on simple models, leading to lower accuracy (around 65%) in complicated or noisy settings.

Emergence of Neural Networks and Hybrid Techniques

Deep learning changed NLP and command recognition in the early 2010s. RNNs and LSTM networks showed better skills for understanding sequences. A key study by Graves et al. in 2013 used LSTMs for speech recognition, hitting 80-85% accuracy. However, these models often got

criticized for being hard to understand, which created problems in use cases where gaining user trust was very important.

At the same time, researchers tried combining decision trees with neural networks. Kontschieder et al. 2015 introduced neural decision forests, joining the clarity of decision trees with the smart features of neural networks. This strategy made classification tasks more accurate, sparking interest in its use for NLP.

Neural Decision Trees in NLP

Neural decision trees became popular as researchers looked to balance clarity, accuracy, and efficiency in command models. An important study by Frosst and Hinton in 2017 brought forward "Distilled Neural Networks into Soft Decision Trees," using gentle decision boundaries for better parsing. This greatly improved understanding of natural language commands, showing accuracy rates between 85-90% in structured tasks.

In research for accessibility, Singh et al. in 2018 used neural decision trees for parsing voice commands from users with speech issues. By mixing Word2Vec embeddings with soft decision trees, their system managed speech variations better, reaching an 88% accuracy rate. This was a significant step toward more inclusive command tech.

Recent Advances: Contextual Embeddings and Transformer Integration

With transformer models like BERT and GPT becoming famous, neural decision trees started using contextual embeddings for better parsing. These embeddings enriched semantic understanding, allowing trees to deal with unclear or context-dependent commands better.

An important study by Zhao et al. in 2020 proposed a hybrid model blending BERT embeddings with neural decision trees. This model achieved top performance, with an accuracy of over 92%, by using BERT's contextual power and tree clarity. This method proved very useful in situations where grasping user intent is very important.

The most important breakthrough in the topic was Zhang et al. (2021), which combined neural decision trees with attention mechanisms. The model integrated attention-based embeddings to dynamically weight relevant parts of a command with respect to input transcription noise leading to parsing accuracy improvements in noisy environments. The system achieved an accuracy of 93% and showed resilience to real-world accessibility tests.

Future Directions and Current Trends

Latency bounds in execution are pursued with current state-of-the-art neural decision trees for command recognition which integrate multimodal data. VLTs are researched by researchers for the task of parsing and executing commands involving visual and textual inputs, enhancing accessibility for users with disabilities even further. Also, reinforcement learning is advancing neural decision trees to adapt to user types of preferences making experiences much more tailored.

Future research is likely to address challenges such as:

Using this work to reduce false positives in ambiguous command scenarios.
Promoting robustness to distinct linguistic, and speech patterns.
Holistic command recognition integrating multimodal inputs (e.g., gestures and voice).
To enhance scalability and efficiency for deployment in real-time systems.

This thesis traces the evolution of command recognition models from rule-based systems to interpretable, hybrid systems such as neural decision trees. These models combine the strengths of deep learning and traditional decision trees to propose a promising solution to help parsing and execution in accessibility-focused applications. As continued progress in contextual embedding, transformer integration, and multimodal processing continue to make their way into the public domain, we anticipate that neural decision trees will become an important player in tomorrow's command recognition systems, bridging the accuracy, efficiency, and user trust.

Methodology

Step 1: Data Generation

We started by generating a synthetic dataset resulting from applying Llama 3.1-8B-Instruct to evaluate the applicability and robustness of neural decision trees in natural language processing tasks. This synthetic data simulated user commands as they would happen in accessibility-focused scenarios particularly common service apps that would employ these models. Each datapoint had a text-based command and an associated action label.

The rationale for synthetic data generation was twofold:

1. Real-world variability: The cases we consider were synthetically generated so that variability in linguistic structures could be introduced, including variations in syntax and phrasings.
2. Addressing data scarcity: As a result of the relative scarcity of high-quality, public datasets crafted for accessibility to command recognition, synthetic generation was a pragmatic solution.

The labeled command action pairs were represented by the dataset with 5000 records.

Step 2: Data Scrambling

We used data scrambling, as a means of reproducing the real-world challenges of collection, noise, ambiguity, and input variability. In this step, lexical, structural, and semantic noise for data was controlled. The scrambling was run in Python with the NLTK WordNet library replacing words with that word's synonym (30% probability) and adding extraneous noise (20% probability).

The scrambling operation can be formalized as:

$$Scrambled\ Command\ =\ T(C)\ =\ Shuffle(Synonyms(C)\ +\ Noise\ (C))$$

Where:

- C *is the original command*

- $T(C)$ *represents the transformation applied to* C
- $Synonyms(C)$ *maps each word in* C *to a synonym with a probability of* 0.3
- $Noise(C)$ *introduces random extra words or character level perturbations with a probability*
- $Shuffle(x)$ *reorders a chunk of the reordered text*

Step 3: Feature Vectorization

The text data was converted into numerical form using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization which captures the importance of words in each command relative to the entire dataset, ensuring that models could leverage key linguistic features for classification.

Mathematically, TF-IDF for a term t in a document d is defined as:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t)$$

where:

- $TF(t, d) = \frac{Frequency\ of\ t\ in\ d}{Total\ terms\ in\ d}$
- $IDF(t) = log(\frac{N}{1 + DF(t)})$

N *is the total number of documents and* $DF(t)$ *is the number of documents containing* t.

This transformation produced a numerical matrix where each row represented a command, and each column represented a word's TF-IDF value.

Step 4: Neural Decision Tree Configuration

We implemented a Neural Decision Tree (NDT) model leveraging a simple yet powerful architecture designed for interpretability and accuracy. The NDT was designed as follows:

1. Input Layer: Accepted the TF-IDF vectorized inputs

2. Hidden Layer
    a. A fully connected layer with 256 neurons and ReLU activation (The 256 neurons is optimum, allowing the model to work with the diversity of the input data while not being too large)
    b. Dropout 30% to prevent overfitting, making sure the model can generalize the training to future data rather than memorize it
    c. A fully connected layer with 128 neurons and ReLU activation (The 128 neurons help reduce the complexity as after the processing from the first layer, identifying specific patterns from generalizations)
    d. Dropout (20%) to again avoid overfitting while leveraging more of the model's neurons to ensure accuracy closer to the output
3. Output Layer: A fully connected layer with softmax activation function to output class probabilities, allowing for multi-class classification

It can be represented as:

$$P(y_i|\,X) = \frac{exp(Z_i)}{\sum_{j=1}^{C} exp(Z_i)}$$

Where (Z_i) is the logit for class i, C is the total number of classes, and $P(y_i|\,X)$ represents the probability for the i-th class for the input X.

The architecture was implemented using TensorFlow's Keras API, and the model was compiled with the Adam optimizer ($n = 0.001$) and sparse categorical entropy loss.

$$\zeta = -\frac{1}{N}\sum_{i=1}^{N} log(P(y_i|\,X))$$

Where N is the batch size, X is the i^{th} input, and y_i is the true label.

Step 5: Training and Validation

The dataset was split into training (80%) and testing (20%) subsets. The NDT model was trained using a batch size of 32 for up to 20 epochs, with early stopping implemented to halt training if the validation loss did not improve for three consecutive epochs.

Accuracy: $Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$

Precision: $Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$

Recall: $Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$

$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

ROC- AUC : Area under the Receiver Operating Characteristic curve, measuring the trade-off between true positives and false positive rates

Step 6: Comparative Analysis

The NDT's results were compared against:

1. Multinomial Naive Bayes (MNB): Known for simplicity and baseline comparisons in NLP tasks.
2. Decision Tree Classifier (DT): Representing traditional interpretable models.
3. Random Forest (RF): Leveraging ensemble techniques for improved performance.

Results:

The provided results showcase the performance of four classification models—Multinomial Naive Bayes, Decision Tree, Random Forest, and Neural Decision Tree —evaluated across five key metrics: All Accuracy, Precision, Recall, F1-Score, and ROC-AUC. Each metric is analyzed for each model, and the performances are compared, strengths are noted and weaknesses explored, alongside explanations of reasons behind these outcomes.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
Multinomial Naive Bayes	89.62	91.44	89.62	89.05	99.64
Decision Tree	83.51	84.08	83.51	82.75	91.64
Random Forest	93.35	94.34	93.35	93.15	99.55
Neural Decision Tree	93.62	94.88	93.62	93.64	99.83

Discussion:

1. Accuracy

Definition: Exactly the proportion of true positives and true negatives among the predictions that were made.

- Multinomial Naive Bayes: 89.63%
- Decision Tree: 83.51%
- Random Forest: 93.35%
- Neural Decision Tree: 93.62%

Analysis:

Neural Decision Trees (93.62%) and Random Forest (93.35%) emerge as the most accurate models, able to accurately predict most of the commands, reflecting modelling of complex relationships within these models.

Multinomial Naive Bayes (89.63%) is generally accurate showing that its probabilistic approach performed well with this dataset, however simplistic. It likely struggled with more complex structures as it assumed feature independence which can oversimplify many commands.

The Decision Tree (83.51%) has the lowest accuracy and most likely suffered from underfitting, lacking the depth of ensemble models to work with the command-action based dataset.

Insights:

The two stronger methods generally perform better because ensemble methods (Random Forest, Neural Decision Tree) can capture complex relationships and minimize variance. The other methods suffer from a lack of complexity resulting in oversimplifications and underfitting, showing the accuracy of ensemble models for a command recognition problem.

---

2. Precision

Definition: How many selected items are relevant divided by the total (i.e., how many predictions are true positive).

- Multinomial Naive Bayes: 91.44%
- Decision Tree: 84.08%
- Random Forest: 94.34%
- Neural Decision Tree: 94.88%

Analysis:

The Random Forest and Neural Decision Tree models emerge with the highest precision at 94.34% and 94.88% respectively with the the Multinomial Naive Bayes model achieving 91.44% and the lowest precision was with the Decision Tree at 84.08% correlating to the highest rate of false positives.

Insights:

The higher precision is present in the ensemble models and this indicates their resistance to over-prediction while correctly identifying positive instances. This likely originates from the complex structure of these models as compared to the simplified nature of the Decision Tree which was less effective, generating false positives from the lower processing.

3. Recall

Definition: This ratio of true positive predictions to all actual positives (how many relevant items we choose) is known as Recall.

- Multinomial Naive Bayes: 89.63%
- Decision Tree: 83.51%
- Random Forest: 93.35%
- Neural Decision Tree: 93.62%

Analysis:

The results of both the Neural Decision Tree and Random Forests show over 93% which is more than enough as they provide the majority of actual cases. Recall in the multinomial Naive Bayes model is at a solid 89.63%, to achieve positives without negligible false negatives. However, recall is lowest at 83.51% for the Decision Tree indicating that it clearly fails to identify an important portion of actual positives.

Insights:

In applications where missing positives are costly (e.g., medical diagnoses), high recall of ensemble models is critical: most positive instances should be identified. In some cases, the Decision Tree's lower recall may require redrawing its complexity criteria or pruning strategy to more accurately capture the positive class.

4. F1-Score

Definition: Precision and recall harmonic mean – a balance between both.

- Multinomial Naive Bayes: 89.05%
- Decision Tree: 82.75%
- Random Forest: 93.15%
- Neural Decision Tree: 93.65%

Analysis:

Top Performers: High F1 scores above 93%, show that Random Forest and Neural Decision Tree achieve a good precision versus recall trade-off.

Moderate F1-Score: Good but not top-tier balance for Multinomial Naive Bayes: 89.05%.

Lowest F1-Score: Overall the Decision Tree underlines its poorer performance when it comes to both precision and recall with 82.75%.

Insights:

Ensemble models outperform all others in the balance of false positives and false negatives according to the F1 scores. Multinomial Naive Bayes performs well, but behind ensemble methods, and it probably is because multinomial naíve Bayes assumes independence in the feature domain.

5. ROC AUC (Receiver Operating Characteristic - Area Under Curve)

Definition: It measures the model's discrimination of classes at all threshold levels. The greater the ROC AUC the more separable they are considered to be.

- Multinomial Naive Bayes: 99.64%
- Decision Tree: 91.64%
- Random Forest: 99.55%
- Neural Decision Tree: 99.83%

Analysis:

Highest ROC-AUC: The Neural Decision Tree achieves the best at 99.83% indicating exceptional class separability.

Close Competitors: Both Multinomial Naive Bayes and Random Forest achieve more than 99% discriminative capability.

Lower ROC-AUC: Still decent, still lower than the others is Decision Tree at 91.64%.

Insights:

This indicates that, with high ROC-AUC across models (particularly ensemble ones), the models are highly effective at ranking positive instances higher than negative ones. Maybe the Decision Tree's lower ROC-AUC is because it cannot capture nuanced class boundaries, and, thus, it overall is not discriminative enough.

Comparative Implications and Insights.

Ensemble vs. Single Models:

Across all metrics, we consistently see that Random Forest and Neural Decision Trees (both ensemble methods), outperform single models such as Decision Tree and Multinomial Naive

Bayes. Combining multiple models offers advantages in ensemble methods which reduce variance, generalize better, and characterize complex patterns.

Model Complexity and Overfitting:

Based on the Depth and Pruning strategy, the Decision Tree suffers from overfitting or underfitting. Lower performance metrics indicate possible underfitting or incorrect hyperparameter settings. This suggests that Neural Decision Trees probably incorporate neural network mechanisms to improve their ability to model intricate relationships and are thus better in performance.

Bias-Variance Tradeoff:

It is explained why despite simplicity, multinomial Naive Bayes achieves such strong performance by being a low variance, but biased algorithm assuming feature independence. Ensembles of base learners find a good compromise between bias and variance.

Applicability of Models:

Multinomial Naive Bayes is particularly suited for text classification and scenarios with discrete features. Its high performance here suggests the data might align well with its assumptions. Neural Decision Trees offer flexibility and robustness, making them suitable for diverse and complex datasets.

ROC-AUC Insights:

Given that ROC-AUC is nearly 100% for all but the Decision Tree, it indicates that the classes are well-separated, and the models are highly effective in distinguishing between them. This high ROC-AUC also suggests that, even if accuracy is slightly lower, the models are making meaningful distinctions between classes.

Conclusion

The experimental results indicate that ensemble models—specifically the Neural Decision Tree and Random Forest—excel across all evaluated metrics, demonstrating superior accuracy, precision, recall, F1-score, and ROC-AUC. Multinomial Naive Bayes also performs

commendably, especially given its simplicity, while the Decision Tree shows potential for improvement. These insights not only highlight the strengths of ensemble approaches in classification tasks but also underscore the importance of model selection and optimization based on the specific characteristics and requirements of the dataset at hand. In the avenue of command recognition models, this demonstrates that a neural decision tree approach could be a stronger method for smaller businesses to integrate the model without having to seek rights to copy-righted ensemble models like Random Forest.

Future Considerations:

Future work can be directed towards several avenues for subsequent improvement in the efficacy and inclusiveness of neural decision trees in natural language command recognition for accessibility. First of all, we still need to push forward model interpretability — this is important because appropriate neural decision tree architectures will help build more trusty user interfaces for accessibility applications and help debug them. Furthermore, increasing training dataset diversity and representativeness is necessary to ensure models can handle a wider variety of linguistic (variation) and speech (impairment). Combining multimodal inputs, for instance, voice commands alongside gesture recognition or even visual cues can give rise to more holistic and intuitive user interfaces to meet the needs of all users. For actually deploying these systems to your everyday environments in which latency can affect the user experience, these models will be optimized for real-time processing. For instance, accessibility can be further increased through personalization through adaptive learning techniques, this is where the models' responses are tailored to the individual user behavior, or, for example, preferences. In addition, issues of training data bias and fair technology that is fair to all user groups will be essential to building equitable technologies. Moving forward, we explore more sophisticated hybrid models combining the strengths of neural decision trees alongside many deep learning paradigms (transformers, reinforcement learning, etc) for unique contributions to parsing accuracy and execution reliability. Ongoing collaboration with end-users and accessibility experts shall, in addition, contribute to the realization of future developments being consistent with real-world needs and challenges of people using these technologies to develop more efficient and inclusive command recognition systems.

Bibliography:

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *35*(3), 400–401. https://doi.org/10.1109/TASSP.1987.1165128

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286. https://doi.org/10.1109/5.18626

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, *4*, 77–90. https://doi.org/10.1613/jair.279

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 6645–6649). https://doi.org/10.1109/ICASSP.2013.6638947

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulo, S. R. (2015). Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 1467–1475). https://doi.org/10.1109/ICCV.2015.171

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv Preprint*. https://arxiv.org/abs/1711.09784

Yang, Y., Morillo, I. G., & Hospedales, T. M. (2018). Deep neural decision trees. *arXiv Preprint*. https://arxiv.org/abs/1806.06988

Tanno, R., Arulkumaran, K., Alexander, D. C., Criminisi, A., & Nori, A. (2019). Adaptive neural trees. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1807.06699

Zhou, Z.-H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 3553–3559). https://doi.org/10.24963/ijcai.2017/497

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197–227. https://doi.org/10.1007/s11749-016-0481-7