Resolving Ambiguity in Hedge Detection by Automatic Generation of Linguistic Rules*

Tracy Goodluck Constance[1][0000−0002−7172−2735], Nikesh Bajaj[1][0000−0002−3361−0118], Marvin Rajwadi[1], Harry Maltby, Julie Wall[1][0000−0001−6714−4867], Mansour Moniri[1][0000−0002−5564−0692], Chris Woodruff[2], Thea Laird[2], James Laird[2], Cornelius Glackin[3][0000−0001−5114−6403], and Nigel Cannings[3]

[1] University of East London, UK
[2] Strenuus Ltd., London, UK
[3] Intelligent Voice Ltd., London, UK
T.GoodluckConstance@uel.ac.uk

Abstract. An understanding of natural language is key in order to robustly extract the linguistic features indicative of deceptive speech. Hedging is a key indicator of deceptive speech as it can indicate a speaker's lack of commitment in a conversation. Hedging is characterised by words and phrases that display a sense of vagueness or that lack precision, such as *suppose, about*. The identification of hedging terms in speech is a challenging task, due to the ambiguity of natural language, as a phrase can have multiple meanings. This paper proposes to automate the process of generating rules for hedge detection in transcripts produced by an automatic speech recognition system using explainable decision tree models trained on syntactic features. We have extracted syntactic features through dependency parsing to capture the grammatical relationship between hedging terms and their surrounding words based on meaning and context. We tested the effectiveness of our model on a dataset of conversational speech, for 75 different hedging terms, and achieved an F1 score of 0.88. The result of our automated process is comparable to existing solutions for hedge detection.

Keywords: Hedge Detection · Resolving Ambiguity · Linguistic Indicators · Linguistic Cues

1 Introduction

There is a real need for the use of decision support systems to help assess the credibility of speech, for example in insurance claims. The UK insurance market has felt the impact of a year-on-year surge in insurance fraud, with an increase

* Supported by Innovate UK National Project: 'Automation and Transparency across Financial and Legal Services: Mitigating Risk, Enhancing Efficiency and Promoting Customer Retention through the Application of Voice and Emotional AI', Grant no. 104817.

in the volume of under investigation. There have been an estimated 3.8 million incidents of insurance fraud in the year ending March 2019, resulting in an increase of 17% from the previous year [2]. The use of decision support systems that can reliably identify the deceptive cues in speech to assist contact centre agents is needed [1]. A credible insurance claim applicant would be expected to respond with a smooth flow of information when answering a query. Any deviation from the expected denotes linguistic sensitivity in the narrative, and certain linguistic features can reveal deceptive traits.

The main challenge for hedge detection is to optimise the effectiveness of keyword spotting by eliminating ambiguous terms. It can be misleading to simply identify all hedging terms in a phrase without correctly considering their use in context. For instance, in the first phrase below, **'suppose'** should be identified as a hedging term, but not in the second phrase [14].

Sentence 1: I *suppose* the package will arrive next week.

Sentence 2: I'm *supposed* to call if I'm going to be late.

There is currently no publicly-available corpus for conversational language for hedge detection. As hedge detection is domain dependent, using an existing out-of-domain corpus will not produce significant results [3]. Therefore, as part of this research, we have created a Conversational Dataset to detect Linguistic Markers, namely the CDLM corpus, from movie and TV scripts sourced online and labelled by linguistic experts for both True and False hedging. Annotation of this corpus is an on-going process with a range of different linguistic markers, of which hedging is one of them. The 75 hedging terms used in this paper are a subset of a broad list of hedging terms that were determined by linguistic experts, from which they were able to annotate enough data for 75 hedging terms.

In this study, we propose an automated approach to hedge detection in speech transcripts by performing disambiguation to filter out irrelevant hedge terms in the conversation. We trained syntactic features with explainable decision trees to visualise the working process of the model and to automate the process of generating the rules for hedge detection in order to derive contextual information. Syntactic features are extracted through parsing, also known as syntactic analysis. Parsing exposes the grammatical structure of sentences and how words are related in a sentence, using the knowledge given by the part-of-speech (POS) tag of a word, such as a noun tag or a verb tag. Semantic and syntactic analysis play an important role in the process of dependency parsing as the semantics verify whether the syntactic features generated by the parse tree follow the rule of language [22]. *"The dog crossed the road."* is grammatically correct. But if it were reversed it would be in this form: *"The road crossed the dog"*, which is grammatically correct but semantically incorrect. In Linguistics, it is very easy for humans to make sense of words and understand the grammatical construction and relationship between these words, whereas in computational linguistics, the machine must learn and encode a pattern introduced by rules generated by the process of dependency parsing. Specific syntactic features have been extracted from the CDLM corpus and fed into a decision tree classifier to determine the essential features to select for optimising hedge detection. Decision trees are

adaptable for problem-solving due to their transparency in decision-making and specificity in assigning values to an outcome [4].

The rest of this paper is structured as follows: In Section 2 we provide a literature review on the state of the art; in Section 3 we describe the annotated corpus used in this work and give a detailed description of the methodologies employed; in Section 4, we present and discuss the results; finally, we outline our conclusions and plans for future work in Section 5.

2 Literature Review

Hedge detection has inspired much work during the last decade. However the work that has been conducted in this field have mostly adopted techniques that are not explainable or constructed using manually crafted rules. The work has principally been undertaken within the Biomedical field using the Computational Natural Language (CoNLL) 2010 framework, which represents a shared task of identifying hedging terms and their linguistic scope in natural language [6]. The shared task tackles it as a sentence labelling classification problem or as a word by word token classification problem which follows an IOB format (Inside, Outside, Beginning) for tagging the scope of the hedging term [7].

The BioScope corpus provided in the framework contains biomedical abstracts and articles annotated with hedging information [8]. Most of the best performing solutions within the CoNLL 2010 shared task framework adopted a sentence labelling approach based on Conditional Random Fields (CRFs) or Support Vector Machine (SVM)-based Hidden Markov Models (HMMs) using the biomedical corpus [7]. The top ranked system for the sentence labelling classification problem adopted commonly used features such as (words, lemmas, POS, and chunks of neighbouring words) and machine learning techniques [7]. A comparative empirical study was completed of four different machine learning approaches to the problem, selecting CRFs, SVMs, K-Nearest Neighbours (KNNs) and Decision Trees. The addition of classification labels for previous words resulted in the SVM classifier yielding the best performance with an F-measure of 0.8682. SVM-based classifiers were employed to tackle the task as a disambiguation problem [9], by using syntactic features to identify hedge terms in and out of context in a sentence, achieving an F-measure of 0.8664, which indicates an improvement of 1.23% in comparison to their initial version of a hedge detection system implemented with a maximum entropy (MaxEnt) classifier [10]. An incremental approach combining the results of a CRF and an SVM-based HMM achieved an F-measure of 0.8636 [11]. A CRF-based and syntactic pattern-based system was also implemented [12], by exploiting the synonym features from WordNet with an F-measure of 0.8632. Using CRFs, a greedy-forward feature selection approach was adopted to boost performance and obtained an F-measure of 0.8589 [13] .

Some recent work in hedge detection uses informal language corpora for hedge detection. For example, the first social-media corpus of 326,747 posts was collated from Twitter's API relating to the 2011 London Riots [3]. They achieved an

F-measure of 0.8212 by employing SVMs to explore the effectiveness of using different features such as n-grams, content-based (regrouping similar information, e.g the geo-location of users in a tweet), user-based (user profiles and followers distributions) and features specific to Twitter. N-grams are a combination of co-occurring words within a phase, for example: "The boy is kind"

- Unigrams are the unique words in a phrase e.g "The", "boy", "is", "kind"
- Bigram is the combination of two words e.g "The boy", "boy is", "is kind"
- Trigram is the combination of three words e.g "The boy is", "boy is kind".

Unfortunately, this annotated corpus is not publicly accessible. Similarly, Amazon's Mechanical Turk was used to annotate a corpus of forum posts from the 2014 Deft Committed Belief Corpora [14]. They annotated the corpus with hedge information and intend in the future to make it publicly available through the Linguistic Data Consortium. They tackled the hedge detection approach as a disambiguation problem, but used a rule-based approach instead of machine learning to manually construct a set of rules to disambiguate potential hedging terms. They adopted hedge detection as a pre-processing technique to extract non-ambiguous hedging features to improve the performance of a committed belief tagging task, another area of research that is closely related to hedging but focuses on the extraction of propositions in the text to determine what the speaker believes [15]. They achieved an F-measure of 0.7270 using the features extracted from their rule-based hedge detection.

3 Methodology

The CDLM corpus was annotated by two behavioural linguistic experts, who manually labeled each spoken utterance with target labels for the identification of hedging terms, both in (True) and out of (False) context. The agreement between the linguistic experts was calculated using Cohen's kappa coefficient, we achieved a substantial inter-annotator agreement of 0.8055. After the experts reviewed the data independently, they came together to re-examine the discrepancies and provided a final decision. The corpus consists of 9,011 spoken utterances, and each utterance consists of one or more hedging terms from a set of 75 unique hedging terms; 59% of the utterances are True hedging, and 41% are False hedging. It is worth mentioning that an utterance labelled as False hedging still contains a hedging keyword, but it is not spoken in the context of hedging. The corpus was split into 70% for the training set and 30% for the testing set, evenly distributed across the sample size of hedging terms.

The proposed approach is initially based on work by [14], who manually constructed a set of rules for detecting ten hedging terms. In our approach, we implemented a machine learning model using Decision Trees, that leverages context to resolve ambiguities for hedge detection in spoken utterances, i.e. it will automatically detect True hedging in speech. Syntactic features capture the context of the utterance through dependency parsing. Using the CDLM corpus and the corresponding expert labels, a Decision Tree model was trained for each

of the 75 hedging terms to generate the rules. The architecture of the Hedge Detection Model in this work can be seen in Figure 1.

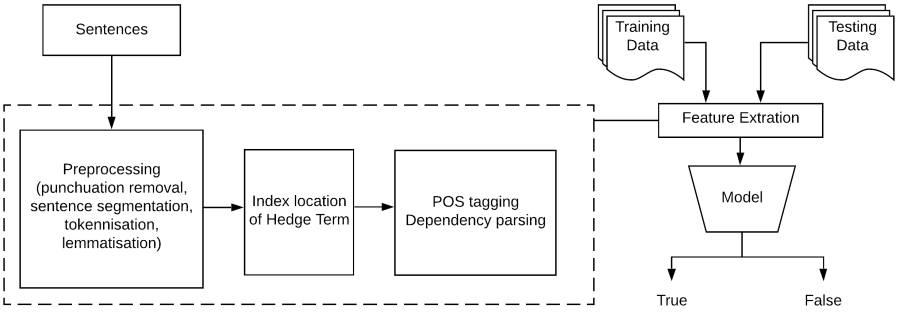

Fig. 1. Architecture of the Hedge Detection Model

3.1 Pre-Processing and Feature Extraction

spaCy's English language model was employed for pre-processing and extracting features from the corpus [16]. Punctuation was removed and the spoken utterances were converted to lower-case to simulate the output produced by an automatic speech recognition system before inputting into our Machine Learning model. Each utterance iterates through spaCy's NLP (Natural Language Processing) function, which tokenises the text to generate a Doc object, making it accessible for different tasks downstream in the pipeline. The pipeline holds all the information about the tokens, their linguistic features and their relationships. The Doc object then segments individual sentences and creates a sentence collection called a generator object.

Lemmatisation captures the different variants of the hedge terms in the spoken utterances, regardless of tense, e.g. appear, appears, appeared, appearing, all come under the root form of "appear". It converts a word to its root form by considering the context of the word before transforming it through a prior knowledge POS. spaCy determines the POS tag beforehand and assigns the corresponding lemma, e.g. identifying the base form of "running" to "run". The index location of the hedging term was retrieved in each spoken utterance, facilitating the task of feature extraction and context-based analysis. Dependency parsing was used to extract POS and syntactic dependency tags, explained further in the next section. Three different window sizes (2, 3 and 4) were used, based on the location of the hedging term in the spoken utterance, i.e. neighbouring words that precede and proceed the hedging term. Ultimately, the extracted features are split into training and testing sets to enable the Decision Tree model to learn to select the essential features for automating the rules for hedge detection.

POS tagging assigns tokens in a sentence with their grammatical word categories as POS tags, such as verbs, noun, adjectives, adverbs. The context of a word is determined based on the POS tag. For example, if hedging analysis was based on a Bag of Words approach, the model would not be able to determine the context where 'like' is a verb in the sentence: 'He likes you', and where it is a preposition in the sentence: **'He stood like a statue'**.

A sentence labelled with POS tags, such as: 'Tom tended plants on the roof.' will return: ('Tom', 'PROPN'), ('tended', 'VERB'), ('plants', 'NOUN'),('on', 'ADP'), ('the', 'DET'), ('roof', 'NOUN'), ('.', 'PUNCT') where PROPN is a proper noun, DET a determiner, ADP an adposition and PUNCT is punctuation.

Dependency parsing is a way of generating the syntactic structure of a sentence, also known as syntactic parsing. It generates a parse tree that can capture the relationship between the words in a sentence [17]. Each dependency is comprised of a headword (governor) and its child word (dependent). POS tagging is a precondition for dependency parsing to limit errors. An example of a dependency tree for a spoken utterance that contains the hedging term **'assume'**: **'I assume his train was late'** can be seen in Figure 2, where the arc connects the headword ('assume') to its' dependent words ('I', 'was'). The arc labels (dependency attributes) describe the syntactic relationship between the hedging term and the words that impact its' context ('nsubj', 'ccomp').

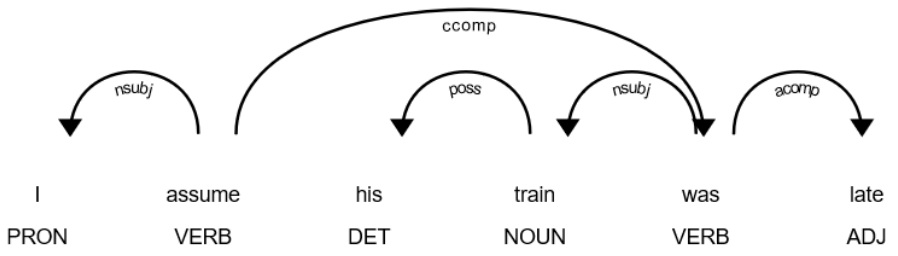

Fig. 2. A dependency tree for a TRUE hedging utterance, hedging term: 'assume'

A set of features were extracted by tagging the location of the hedging term. Hedging terms, both in and out of context, were tagged in order to train the Decision Tree to recognise both True and False hedging. We extracted contextual features, such as POS tags and dependency relation tags concerning the term and its' surrounding words within the range of specific window sizes, determined based on the location (before and after the hedging term).

3.2 Decision Tree Classifier

Decision trees were chosen in this work because of their numerous advantages. They are useful in selecting essential features for making rules and the decisions

made by the classifier are easy to understand, interpret and visualise [4]. Focusing on the explainability of the model was one of our core objectives, as a sound decision support system enables an organisation to promote transparency and accountability [5]. The ability to enhance and facilitate the process of providing an audit trail of the overall decision of the system, leads to a model being adopted and trusted, which enables organisations to be consistent and fair. A system with in-built explainablility will be able to verify predictions, identify flaws and biases in the model, fully understand the original problem, and ensure legislation compliance [18]. When we understand how a model works and can visualise how the input features impact the prediction of a specific class in the model, it makes it possible to investigate why the model makes its prediction and repair flaws [19]. Decision trees provide all these benefits as the information flow of the model is visible, and resolving issues causing errors and biases will be less complicated. A disadvantage of Decision Trees is that bias can be introduced into the model. If the model has been trained on an imbalanced dataset, one class will tend to dominate the other. However, this is a common challenge with all machine learning techniques. A significant disadvantage point to the trade-off between the performance of the model and the explainability of its' predictions [19].

We used the SpKit Signal Processing Toolkit to build our Decision Tree model [20]. The main benefit of SpKit is that for text classification it takes string values as input without having to transform the text into a numerical representation in the form of a vector. The Decision Tree model developed here uses the syntactic features to detect the context of the hedging terms in conversational utterances. The input receives these features, and the model produces a classification output of TRUE or FALSE (whether the detected hedging term in each utterance is 'True' hedging or not).

A Decision Tree uses the Iterative Dichotomiser 3 (ID3) algorithm [4], which constructs a tree representation in response to a given classification question. ID3 uses a top-down greedy approach to build a Decision Tree; it determines the best features to split the data by selecting instances with similar values. A Decision Tree comprises a root node, internal nodes, and leaf nodes, representing the different hierarchies of the tree. The decision is made at the leaf node, which is the last level. ID3 uses entropy and information gain to build a Decision Tree. Entropy represents how the data is split based on the homogeneity of a sample (the entropy is zero if the samples are entirely the same and the entropy is one when the samples are equally divided). Information gain depends on the decrease in entropy after a particular feature has split the data, and Decision Trees use the highest information gain to split or construct a tree.

Figure 3 provides an example of a Decision Tree for the hedging term **'suppose'** which illustrates the different branches of the nodes. The features selected to generate the rule are visualised. This rule validates by navigating from the top-right node to check if a condition is True. If True, navigation continues, otherwise it checks the next node and repeats the same approach. For this specific case, the rule has two options to identify a true hedging term. Firstly, if the tag of a term is a 'VBP' (verb, non-3rd person singular present), and the word that

follows the term has a "to" dependent, then the term is true hedging, otherwise it is not. Secondly, if the tag of a term is not a 'VBP', it will navigate to the left and go to the next node to check if the third word that precedes the hedging term has a "nsubj" (nominal subject) dependent. If true, then the term is true hedging, otherwise it is not. Figure 3 also visualises the rule generated for the hedging term **'assume'**. In this case, the tag features were more important for identifying a hedging term. The rule checks if the tag of a term is a 'VBP' (verb, non-3rd person singular present) and if true, then the term is True hedging, otherwise it is not.

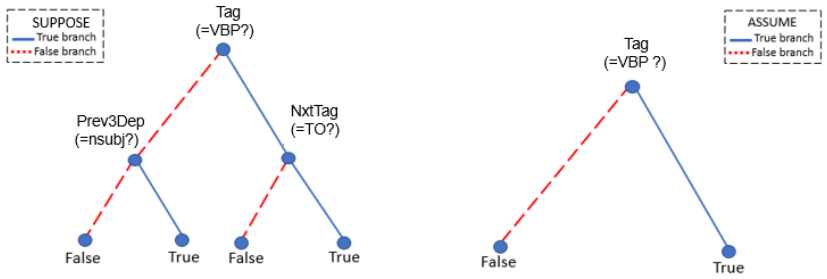

Fig. 3. Example of features selected by a Decision Tree when creating a rule for the hedging terms **'suppose'** and **'assume'**

3.3 XGBoost Classifier

We experimented with XGBoost to investigate its renowned model speed and performance. XGBoost is a popular machine learning method; frequently used on Kaggle for winning competitions as it has outperformed many other gradient tree boosting approaches. XGBoost stands for Extreme Gradient Boosting, and it is a Decision Tree-based ensemble method that uses the gradient descent algorithm for minimising errors in sequential models [21]. This approach performs very well because it boosts weak learners by correcting the residual errors in the sequence.

The performance of the model in this paper can be improved through the many advanced features XGBoost offers for fine-tuning the model. XGBoost only takes numerical values as input, so we had to encode our labels for classification. For this, we used one-hot encoding to transform our features into categorical input variables. Transforming data to one-hot encoding causes data sparsity; however, this was dealt with by an algorithm that XGBoost implements, to handle different types of sparsity patterns in the data. XGBoost produces a final prediction through the output of many trees as it sequentially sums up the

predictions of each tree to improve its performance. When training the XGBoost model, we used the configuration recommendation for parameters given by scikit-learn [21]: learning rate = 0.1, n estimators = 100 (number of trees), max depth = 3 (maximum depth of a tree), min_samples_split = 2 (minimum number of samples required to split an internal node), min_samples_leaf = 1 (minimum number of samples required to be at a leaf node), subsample = 1.0 (fraction of features to be randomly sampled for each tree).

4 Results

We carried out two experiments on our Decision Tree classifiers trained on the CDLM corpus. First, we conducted a baseline experiment to compare the effectiveness of the manually constructed rules developed by [14] and our automated rules generated by Decision Trees, for a subset of the CDLM corpus. We then used the full corpus for the second experiment, to compare the performance of the Decision Tree and XGBoost approaches.

For the baseline experiment, we compared our automated approach against the manual rules from [14], for nine out of their ten hedging rules. The reason for only comparing against nine rules, is that we did not have labelled data for the tenth one. Both approaches were evaluated on a test set of 957 sentences in total from the CDLM corpus. Table 1 shows the number of samples for each hedge term, the manually constructed rules, and the accuracy (%) achieved by both approaches against this test set.

The results show an overall accuracy of 61.85% for the manually constructed rules and 85.31% for the automated rules generated by our Decision Trees. The accuracy of the Decision Trees outperformed the manually constructed rules by 23.45%. We observe there is a significant increase in accuracy for each of the hedging terms, except for the term 'rather' where the manual rule outperforms the Decision Tree by 5%. The reason for this could be that it is a simple rule that requires a Regular Expression approach to find a specific pattern, whereas the Decision Tree was trying to learn the appropriate keyword which was true within the context of the sentence based on the example that we have previously discussed, where *'suppose'* is not always a hedging term. We want to be able to train a model that picks up *'suppose'* when it is a true hedging term. The rules created manually did not generalise well with the CDLM corpus; the reason could be the depth of the syntactic structure of some sentences. Our Decision Trees were trained on both short and long spoken utterances, which gave them the ability to extract grammatical relationships between words in longer utterances. Also, machine learning approaches can learn new rules and adapt to changes, unlike a manual, limited, approach.

For our second experiment, we used the full CDLM corpus of 9,011 sentences and 75 hedging terms to compare the performance of the Decision Trees and the XGBoost classifier for True hedge detection, see Table 2. The best window size for the Decision Tree model was two words before and after the hedging term (± 2) with an F1 score of 0.8896. In comparison, the best window size for the XGBoost

Hedging terms	Sample size	Manually Constructed Rules [14]	Manual %	DT %
About	218	If token t has part-of-speech IN, t is non-hedge. Otherwise, hedge.	75	89
Likely	58	If token t has relation *amod* with its head h, and h has part-of-speech $N*$, t is non-hedge. Otherwise, hedge.	41	92
Rather	100	If token t is followed by token 'than', t is non-hedge. Otherwise, hedge	85	80
Assume	57	If token t has *ccomp* dependent, t is hedge. Otherwise, non-hedge.	79	89
Tend	82	If token t has *xcomp* dependent, t is hedge. Otherwise, non-hedge.	77	80
Appear	38	If token t has *xcomp* or *ccomp* dependent, t is hedge. Otherwise, nonhedge.	58	67
Completely	18	If the head of token t has *neg* dependent, t is hedge. Otherwise, nonhedge.	89	100
Suppose	297	If token t has *xcomp* dependent d and d has *mark* dependent 'to', t is non-hedge. Otherwise, hedge.	47	93
Should	89	If token t has relation *aux* with its head h, and h has dependent 'have', t is non-hedge. Otherwise, hedge.	38	89

Table 1. Classification accuracy (%) for both approaches (Note: IN is the POS tag for preposition)

model was three words before and after the hedging term with an F1 score of 0.8953. Evaluating a range of window sizes contributed to the improvement in both models, indicating that the surrounding words in the spoken utterance give contextual information to the hedging term. XGBoost slightly outperformed the Decision Tree model by 0.0057, even with only limited fine-tuning of the model parameters. Its' strength is due to the boosting algorithm it uses to strengthen weak learners.

	Decision Tree			XGBoost		
Window size	Precision	Recall	F1	Precision	Recall	F1
5 (±2)	0.8917	0.901	0.8896	0.8909	0.8998	0.8895
7 (±3)	0.8905	0.9	0.8884	0.8937	0.9064	0.8953
9 (±4)	0.8908	0.8994	0.8885	0.8881	0.9013	0.8901

Table 2. Classification results of the Decision Tree and XGBoost models

Our results seem very promising compared to other hedge detection solutions available in different domains. Although it is hard to benchmark hedge detection

against similar work due to the lack of publicly available informal language corpora annotated with hedging information. The two closest comparable results are from [14] who achieved an F1 score of 0.727 on their annotated corpus of forum posts, and from [3] who achieved an F1 score of 0.8215 on their annotated corpus based on social media data. Having access to publicly available corpora will provide a stable ground for a fair comparison between different solutions and techniques for hedge detection.

Our approach shows that the performance can be improved by employing more labelled and balanced data, a common theme for machine learning techniques. We employed contextual and positional features to boost the performance of the models, similar to [7, 11]. We also followed the approach of performing disambiguation [9], which [14] also adopted to construct their manual rules by filtering ambiguous hedging terms. Decision trees are a successful solution for this automatic generation of rules, as they simulate the human approach of creating rules for decision making and are easy to interpret.

5 Conclusions

Our findings in relation to the automation of generating rules for hedge detection indicate that we can tackle the problem related to ambiguities when detecting linguistic features in conversational language. Extracting the relationship between potential hedging words or phrases and their surrounding words in a spoken utterance provides the context of whether the term is truly hedging or not. We performed dependency parsing to extract these relationships from utterances, which we used as syntactic features to train our models. The core contribution of this work was using the explainable Decision Tree model to automate the process of generating rules for hedge detection disambiguation. We evaluated the effectiveness of our model on a dataset of conversational speech, with decision models for 75 different hedging terms, achieving an F1 score of 0.88, a comparable result to existing solutions for hedge detection.

In future, we will explore this technique by using word embeddings. Our technique can also be applied to the automatic generation of rules for other types of linguistic markers indicative of deception such as unnecessary explanations (*explainers*) and *memory loss*.

References

1. N.Bajaj et al., "Fraud detection in telephone conversations for financial services using linguistic features," 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), AI for Social Good Workshop, Vancouver, Canada.
2. ONS, "Nature of fraud and computer misuse in England and Wales", [Online]. Available: https://www.ons.gov.uk/aboutus/transparencyandgovernance
3. W. Zhongyu et al., "An Empirical Study on Uncertainty Identification in Social Media Context", ACL, vol. 2, pp. 58-62, 2013.
4. J.R. Quinlan, "Decision trees and decision-making", IEEE Trans. Systems, Man, and Cybernetics, vol. 20(2), pp. 339-346, 1990.

5. M. Rajwadi et al., “Explaining Sentiment Classification, INTERSPEECH, 2019.
6. R. Farkas et al., “The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text”, ACL Conf. Computational Natural Language Learning - Shared Task, pp. 1-12, 2010.
7. S-J. Kang, I-S. King, S-H. Na, “A Comparison of Classifiers for Detecting Hedges”, Int. Conf. U-and E-Service, Science and Technology, pp. 251-257, 2011.
8. V. Vincze et al., “The BioScope corpus: Biomedical texts annotated for uncertainty, negation and their scopes”, BMC bioinformatics, vol. 9(11), pp. 1-9, 2008.
9. E. Velldal, “Predicting speculation: A simple disambiguation approach to hedge detection in biomedical literature”, J Biomedical Semantics, vol. 2(55), pp. 57, 2001.
10. E. Velldal, L. Øvrelid, S. Oepen, “Resolving speculation: MaxEnt cue classification and dependency-based scope rules”, Conf. Computational Natural Language Learning - Shared Task, pp. 48-55, 2010.
11. B. Tang et al., “A cascade method for detecting hedges and their scope in natural language text”, ACL Conf. Computational Natural Language Learning - Shared Task, pp. 13-17, 2010.
12. Z, Huiwei et al., “Exploiting multi-features to detect hedges and their scope in biomedical texts”, ACL Conf. Computational Natural Language Learning - Shared Task, pp. 106 - 113, 2010.
13. X. Li et al., “Exploiting rich features for detecting hedges and their scope”, Conf. Computational Natural Language Learning - Shared Task, pp. 78-83, 2010.
14. M. Ulinski et al., “Using hedge detection to improve committed belief tagging”, W. Computational Semantics beyond Events and Roles, pp. 1-5, 2018.
15. V. Prabhakaran, O. Rambow, M. Diab, “Automatic committed belief tagging”, Coling 2010: Posters, pp. 1014-1022, 2010.
16. B. Srinivasa-Desikan, “Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras”, Packt Publishing Ltd., 2018.
17. V. Teller, “Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition”, Computational Linguistics, vol. 26(4), pp. 638-641, 2000.
18. S. Wachter, B. Mittelstadt, L. Floridi, “Transparent, explainable, and accountable AI for robotics”, Science Robotics 2, vol. 6, 2017.
19. PwC, “Explainable AI Driving business value through greater understanding”, [Online]. Available: https://www.pwc.co.uk/audit-assurance/assets/pdf/explainable-artificial-intelligence-xai.pdf
20. N.Bajaj et al., “SpKit: Signal Processing Toolkit,”, python library, https://spkit.github.io, 2019.
21. J.Brownlee, “XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn,” 2019.
22. T. Dozat, C. D. Manning, “ Simpler but more accurate semantic dependency parsing ”, arXiv preprint arXiv:1807.01396, 2018.