The predictive power of Prosody in Spanish question classification using error-driven learning

Bachelor Thesis
University of Groningen
Artificial intelligence

Author: Daniel van Heuven van Staereling S3585131
SuperVisor: Stephen Jones

August 20, 2023

Abstract

Speech contains more information than just the grammar structure and semantic meaning. The intonation of a speaker gives important information to the sentences meaning. This information is difficult to analyze for a machine. In Spanish, where questions and statements have the same word order, intonation helps distinguish between them. However, context is important in most cases. Therefore, intonation is not the only discerning factor. The lack of context and the noise created by individual differences in speakers make it a difficult problem. This research explores how to develop a process of abstracting pitch to a set of cues that is easier to interpret and what cues to use. Furthermore, this research makes a model to evaluate the predictive accuracy in classifying questions with only pitch information without contextual information to see how much is possible with only pitch. The machine learning model of choice is error-driven learning because of its interpretability. The model needs to train on the created set of cues to learn the important patterns in speech for classifying questions and statements. The process returns a set of cues. Analysis of the results shows that only a little information is necessary. It is only necessary to look at the first and last change in the pitch to classify questions. The cues used give predictions with a balanced accuracy of 64.9% and an averaged F1-score of 62.6%. The accuracy is not 100%. The reason is that people use context in regular conversation. More research is necessary to explore the effects of adding context.

Contents

1 Introduction

When we want medical robots that can eventually communicate with people, it needs to understand what people say. Making robots understand language goes beyond just factual descriptions of words. Changes in intonation are just as important. Humans can pick up these changes easily. Intonation being important is very apparent in Spanish sentences. In Spanish, there are many cases where two sentences have the same syntax structure, but the meaning is completely different. Intonation can make it either a question or a statement. This difference is easy to recognise for humans, but this might be difficult for a robot because the change in intonation can be small and speakers vary in their use of prosody. We want to see how well a machine can learn patterns in pitch from human speech and then predict correctly whether new sentences are questions or statements. If this is possible with higher accuracy than just random guessing, it is a good indicator that this might be something that can help in the future. Changing intonation in spoken language is one aspect of prosody. This research concerns whether machines can use prosody and find patterns in the specific case of these Spanish questions. The introduction will discuss the specifics of prosody. What is prosody? Why is it important? How are we looking at prosody for this research? Then an important concept called error-driven learning needs to be explained, which is necessary for teaching a machine to "understand" prosody in Spanish questions. Furthermore, the specific research question we will answer is at the end.

1.1 Prosody

Prosody is a process that helps guide meaning using speech. Specifically, the definition for prosody according to the Cambridge Dictionary is: "the rhythm and intonation (= the way a speaker's voice rises and falls) of language" (McIntosh, 2013). There are multiple ways of using prosody to change the meaning of a sentence. Intonation is important. Raising or lowering the pitch in a sentence at a specific moment changes the meaning. An example is in Spanish. In English, syntax structure changes when asking or stating something, which makes it easy to recognize. In Spanish, for certain questions, the syntax structure does not change. Only by changing the intonation is the meaning made clear. The tables below show an example.

Spanish		
Bail-a	con	Jan
Dance-3.PRS.SG.SUBJ	with	Jan

'He/She dances with Jan.'

Table 1: Spanish statement example

Spanish		
¿Bail-a	con	Jan?
Dance-3.PRS.SG.SUBJ	with	Jan?

'Does he/she dance with Jan?'

Table 2: Spanish question example

The sentence "Baila con Jan" can be translated to both "Does he dance with Jan?" or "He dances with

Jan." When writing, there is only a noticeable difference because you can see the question mark. However, the actual meaning comes from the intonation. Prosody is often made clear in writing by punctuation. Another case of prosody is by changing the rhythm. By pausing at the right time, it is possible to put focus on certain parts of a sentence. In written speech, commas constitute the pauses from spoken language. The example below illustrates the difference in not pausing in sentence (1) versus pausing in sentence (2).

(1) The man with the coat is very smart.
(2) The man, with the coat, is very smart.

The comma shows where there is a pause while talking. Sentence 1 uses a restrictive relative clause. The clause "with the coat" specifies something about the man. The second sentence has a non-restrictive relative clause. A restrictive relative clause cannot be removed without the meaning changing. It is essential to the sentence. A non-restrictive relative clause is non-specific and can be removed without a problem. The comma in written language or pause in spoken language shows whether the clause is restricted.

These are only a few examples of how prosody is seen in language. In these cases the meaning changes. The statement becomes a question and the clause changes because of the comma/pause. There are also cases where prosody does not change the meaning. Like the following example, prosody can reflect the emotional state of a speaker which does not change the meaning. Furthermore another example, languages have differences in their rhythm of language. There can be changes in how sentences are stressed, like the difference between English and French [1]. These are both forms for using prosody that showcase where prosody keeps the meaning the same even though prosody changed. Hence there is a contrast between the two examples and the previously mentioned cases where the meaning of a sentence changes by using prosody. Therefore, it is important to differentiate between these cases.

For this research, the focus lies on changes in intonation. As seen in the example for Spanish, the intonation changes the sentence's meaning. In that case, there is a clear difference in meaning. Furthermore, It is possible to look at a speaker's pitch during a sentence, which is the physical representation of intonation. The changes in pitch represent the intonation. This research aims to train a machine learning model to classify sentences as questions or statements using only the pitch. Of course, this is a difficult process. Every person has a different voice. For example, women often have a higher pitch than men. Then there is the problem that speech is analog. A machine must interpret the sound. Hence, when recording speech, it becomes digital as a sound file. The effects of digitalizing mean that noise can occur that is purely a consequence of the recording. These variations make it difficult to find patterns. First, the patterns must be general enough to work on different people. Second, it needs to be able to distinguish between important variations because of pitch change and ignore noise. We need a process to find the important patterns and classify the questions or statements.

1.2 Error driven learning

The goal is to have a machine read a sound file as input and classify it as a statement or question with decent accuracy. A machine learning model is needed to train and return some output for making classifications. The type of machine learning model is critical. This research used error-driven learning. So an explanation of error-driven learning is necessary. Important is, why is error-driven learning chosen over other machine learning models? Furthermore, how does it work in this research, and what are some difficulties connected to prosody?

For machine learning, it is uncommon to hear about error-driven learning models. However, error-driven learning models have been used in many fields for a long time already (for example,(Rescorla, 1972)). Even though they are not mentioned often in machine learning, they are a core part of some AI applications based

[1] https://www.technologia.com/en/blog/articles/mastering-the-rhythm-of-english-the-key-to-sounding-like-a-native-speaker

on neural networks, even in recent applications (for example, (Wu et al., 2016)). Error-driven learning is part of a larger, more complex system in these more modern models. However, error-driven learning models are simple and effective. Error-driven learning is useful for modeling human learning. Usually, researchers prefer a deep neural network for making more accurate predictions. However, error-driven learning helps to model the human learning process as it uses the Rescorla-Wagner model. This model uses the point of view from classical conditioning for modeling human learning (Rescorla, 1972). It looks at how the absence or presence of multiple conditioned stimuli predict an unconditioned stimuli. It can be really usefull for explaining the blocking effect (Kamin, 1969). However, there are certain effects it cannot explain. According to experiments for example, it is easier to learn a conditioned stimuli for the second time. If the stimuli is learned and then extinguished, it is easier to relearn (Napier, Macrae, & Kehoe, 1992). It is not perfect. However, the most important part is that it can discriminate between multiple stimuli to learn the activation of these different stimuli.

Error-driven learning is similar to a fully connected neural network with only an input and output layer. Fig. 1 shows an example of a representation from Hoppe, Hendriks, Ramscar, and van Rij (2022). The input data are called cues, and the output is the outcomes. The cues are discreet values that can either be present or not. Cues are different from features, which are common for neural networks. Features are often continuous values. For example, length is continuous and ranges from 0 to any value. In this case, the cues represent attributes of an animal. The outcome is a dog or rabbit. Another important feature which this example illustrates is that error-driven learning is a discriminative model as error-driven learning uses both cue and outcome competition (Hoppe et al., 2022). Which is important because we need to do discriminate between two categories, statements and questions.

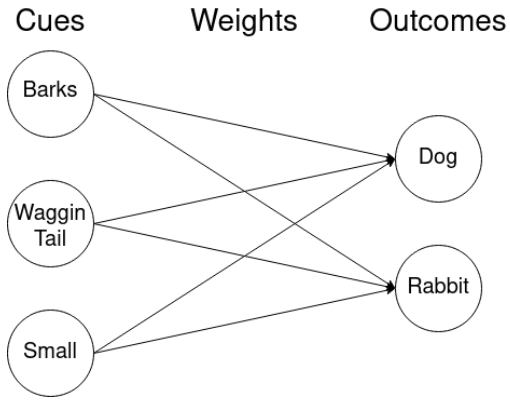

Figure 1: This is a visualisation of what an error-driven learning network looks like. It is an example from Hoppe et al. (2022). It is basically a fully connected neural network without a hidden layer and with different update rule

Error-driven learning is not a complex model. If you have a lot of data and only care about attaining the highest accuracy and not about interpretability then the lack of complexity can be bad, as the lack of complexity makes it more difficult to reach a high accuracy for making prediction by the limitations. To keep it simple usually it requires using less parameters, less layers or other limitations that can improve the accuracy. However, the simplicity of error-driven learning can be useful. The model does not use any hidden layers. Using cues and no hidden layers makes error-driven learning very interpretable.

The weights directly relate from the cues to the outcomes, which makes the model interpretable. If the weight is high, a specific cue predicts a certain outcome. If the weight is low, it does not say anything about an outcome. The updating mechanism shows another good reason for using error-driven learning. Besides its interpretability, error-driven learning does not only look at the present outcome but also the

absent outcome. This means it learns from both positive and negative evidence. This makes learning more efficient as the algorithm can use more of the data and does not need to wait for only positive evidence. Hence, the algorithm does not need as much data. However, how do the weights get updated to accomplish this effect?

updating weights The weights are updated iteratively after being presented by a set of cues. A set of cues are all the attributes of a single object. In the figure, those are barking, wagging tails, and small. All weights are updated. Equation 1 is the equation for updating the weights. Here the letter i means cue i and j is outcome j. The V is the weight, which is at update step t.

$$V_{ij}^{t+1} = V_{ij}^{t} + \Delta V_{ij}^{t} \tag{1}$$

The connection from each cue in a cue set to an outcome updates the weight. The update depends on ΔV_{ij}^{t}, which can be determined based on two different update rules. The first rule is the Delta rule. The delta rule is the formula in Equation 2. There are different cases for when the cue and outcome are present. The update then changes according to the given formula. The different cases are for either positive or negative evidence, if no cue is present nothing happens.

$$\Delta V_{ij}^{t} = \begin{cases} 0 & \text{, cue i absent} \\ \eta(1 - act_j^t) & \text{, cue i and outcome j present} \\ \eta(0 - act_j^t) & \text{, cue i present but outcome j absent} \end{cases} \tag{2}$$

The second update rule is almost the same as the Delta rule. There is a slight change because a few extra hyper-parameters affect the learning process. The update rule is shown in Equation 3 and is called the Rescorla-Wagner learning rule (Rescorla, 1972). Instead of the learning rate η, they use α_i, which can vary by cue and two learning rates, β_1 for positive evidence and β_2 for negative evidence. The other extra parameter is λ, which is in the second case of the learning rule. The regular Delta rule sets λ to 1.

$$\Delta V_{ij}^{t} = \begin{cases} 0 & \text{, cue i absent} \\ \alpha_i\beta_1(\lambda - act_j^t) & \text{, cue i and outcome j present} \\ \alpha_i\beta_2(0 - act_j^t) & \text{, cue i present but outcome j absent} \end{cases} \tag{3}$$

The learning changes based on the order of the cues (Hoppe et al., 2022). For training, the order of the data is random. Combined with a long enough training length, this ensures the weights converge. These steps should result in smaller variations of these effects.

After completing the training and determining the weights, testing is necessary. There is a formula for calculating the activation value for each new data point. The activation for some outcome j at time t is (act_j^t). The formula for the activation value is in Equation 4. The activation is proportional to the activation, because the activation is the sum of the weights. This is what results in the model being able to differentiate the cues that are informative from the general cues.

$$act_j^t = \sum_{x \in cues(t)} v_{xj}^t \tag{4}$$

Problems Error-driven learning has some problems for this research. As mentioned before, pitch is analog. Even after digitalizing, it is still considered a continuous value. Fig. 2 shows the pitch continuity as the blue line in the PRAAT program. Error-driven learning uses cues. Pitch values are not compatible with

cues. That is why it is necessary to do preprocessing so the model can get some cues as input from the pitch. From looking into the literature, there is no method for doing this. However, cues are necessary for error-driven learning. Hence, the methodology requires a process that creates an abstract representation of cues from the pitch. This process needs to make an abstract representation of the pitch that is interpretable and shows the changes in the pitch in a meaningful way.

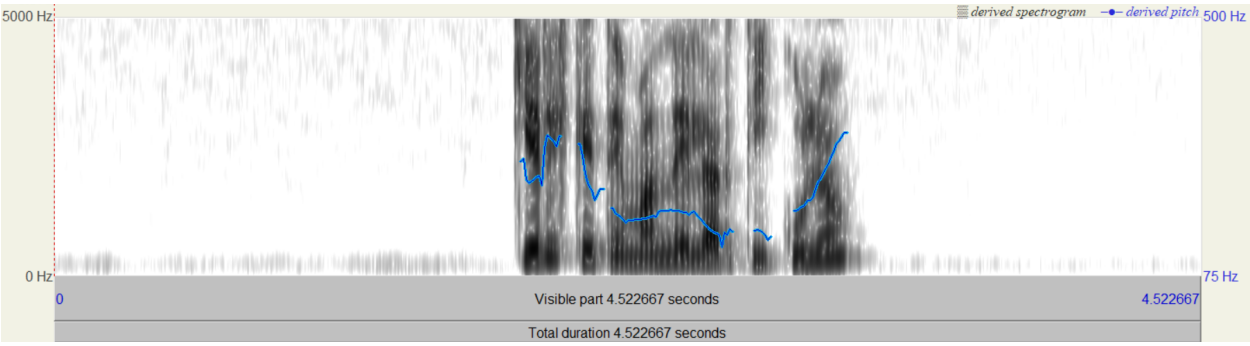

Figure 2: This shows an example of how a sound file looks like when visualised in the program PRAAT. The blue line is the pitch which is in Hz. The value goes from 75 Hz to 500 Hz which is determined based on settings in praat. The pitch is determined based on a process called auto-correlation. The black areas are the spectogram of the frequencies of the speech signal, they go from 0 to 5000 Hz. The spectrogram is determined with the fourier transform method.

1.3 Speech recognition existing methods

Error-driven learning There has yet to be much research on error-driven learning for speech recognition. Arnold, Tomaschek, Sering, Lopez, and Baayen (2017) looked at speech recognition with error-driven learning. The problem is that they use single words as input instead of complete sentences, as is the case for this research. They use a method that uses numbers for the pitch in discrete form as cues. They take the mean of different chunks, but it still results in many possible cues and, therefore, in many parameters. The amount of data must account for that, which may not be the case for this research.

Furthermore, there is more variation in a complete sentence than there is for words. Even though they concluded that phones are less important than once thought, such a conclusion might not translate to complete sentences. Lastly, this research has a second focus. This research tries to make the model interpretable, so the cues can be used for making conclusions about the patterns for both questions and statements. Arnold et al. (2017) uses numbers which makes interpretability for finding these patterns more difficult. However, this requires a different method for this research.

Other algorithms Error-driven learning is not the only available machine learning algorithm. There are others. Much research is into speech emotion recognition (Panda, Jena, Panda, & Panda, 2023), (Paul, Bera, Dey, & Phadikar, 2023). Emotion recognition is most similar to the research performed for this thesis as it does not concern itself with the meaning of the words. Multiple feature types were analyzed, including MFCC or ZCR, for example.

Other interesting features include a sound signal's speaking tempo or duration length. Another paper used these features, among others, to predict when a machine is more likely to make a wrong prediction (Litman, Hirschberg, & Swerts, 2000). These features could affect the problems that occur when looking at changes in speech. Speaking tempo helps for these speech signals as it adds extra information for why there are fewer variations in the speech signal.

Formants are valuable features when trying to look at speech. It is especially valuable when trying to distinguish speakers. The first two formants determine vowel quality. The most important information comes from the second and third formants as they change between speakers. One paper tried to predict individual speakers and used formants among other features (Agrawal, Shruti, & Krishna, 2010). However, this research only cares about pitch, which is the zero formant. The other formants cannot be used for this research as it does not contribute to the topic.

In another paper, they looked at usable features for gender identification (Shagi & Aji, 2022). Their paper used a bunch of features. Most of them manipulate the pitch in different ways. The features they used were all continuous values. One of the features they used was the absolute sum of changes (ASC). This used equation 5. This feature is interesting as it shows whether there is much change in the speech signal. However, it is still challenging to work with since it is continuous.

$$ASC = \sum_{i=1}^{n-1} |p_{i+1} - p_i| \tag{5}$$

What is apparent from the papers concerning other machine learning algorithms is that they use continuous values. Using continuous values is easier for most machine learning classifiers. As discussed before, error-driven learning works differently. It is dependent on the algorithm what features are helpful. If the research used other algorithms, these would be good features.

1.4 Research goal

As mentioned, it is difficult for machines to recognize prosody. If we only give pitch information to a model, can it learn to recognize patterns in the data and make correct predictions? It is not possible to generalize multiple prosody processes across different languages. Hence, this research considers Spanish for questions and statements. Questions or statements are binary, and as Spanish does not have the syntax change, this should account for a more recognizable difference in the pitch. These reasons make Spanish an excellent example for classifying a part of prosody. The main research question I will answer is:

"How accurately can the error-driven learning algorithm be trained in predicting whether a sentence in the Spanish language is a question or a statement with only information from the pitch?".

Furthermore, this research is looking at an extra research question regarding the part of speech that is most important for recognizing the type of sentence for an error-driven learning model. The subquestion is:

"Which part of the sentence is most informative in making predictions for an error-driven learning model?".

Answering the sub-research question requires multiple methods. Error-driven learning already exists, so existing methods can be used to train the model. However, no preprocessing method exists that is also interpretable for turning the pitch into the necessary cues for sentences. Hence, a sub-question needs to be answered, "How can pitch, which is continuous, be turned into an abstract representation of cues that can also be used to identify the most important part of the sentence for making the prediction?". The second research question requires extra work to determine the most essential part of the sentence, which requires an analysis after training the cues.

2 Methods

First, we need to discuss the data distribution between questions and statements. Then the method of creating the pipeline. The method consists of making a pipeline that performs several steps. Some steps came from existing papers, and other steps in the process are new from exploring by trial and error. The first step is to read a sound file as input and extract pitch. The program that performs the extracting is openly available. Second, as mentioned in the introduction, this research is concerned with making a preprocessing method that transforms the pitch into an abstract representation to get cues for error-driven learning. There is no available method in the existing literature. Hence, these came from testing out different things. The third step is the training process. The training and testing came from existing methods of error-driven learning (Hoppe et al., 2022). The created method of cues provides the input for the error-driven learning model, which can train on the data. Lastly, the trained model must make predictions on a test data set and the predictions are analyzed.

2.1 Data

The data consists of a total of 5739 sound files of native Spanish speakers from the Argentinian dialect. The sound files have different speakers. Of the sound files, 1818 are recorded by males and 3921 by females. The data contained corresponding lists with the annotation of the sound files. Some sentences consist of a statement and a question indicated by the Spanish upside-down question mark not being at the start. This research did not consider these types of sentences as they act differently from other questions. Leakage should not occur. Hence, splitting the data into a train and test set is necessary. After splitting, the train data consisted of 4513 sound files, 1133 questions, and 3380 statement sentences. The test set had 1128 sound files, 246 questions, and 882 statement sentences. The data was retrieved from Guevara-Rukoz et al. (2020).

2.2 Preprocessing method

There are several steps for preprocessing. It starts by extracting pitch from the sound files. For the following steps it is important to only use the necessary information to avoid silences and take a range to avoid unnecessary information. Furthermore, it is important to normalize and eliminate gaps in the sound by interpolating. The third step in preprocessing is to eliminate as much noise as possible. There is no clear existing method for deleting noise like this, so this requires trial and error and choosing a method that gives a smooth representation. The fourth step is to transform the signal into an abstract representation. The continuous signal cannot be transformed directly into cues. Some abstract representation is needed first. Just as for the elimination of noise, there is no existing method. Hence, this requires the creation of such an abstraction process. The last step is to transform this abstraction into cues for the model.

Praat for pitch extraction The first step is to extract the pitch from each sound file. The program PRAAT is common for handling sound files in analyzing speech. PRAAT is a program that can extract pitch. The extraction process can be automatized by writing scripts in a specific language for PRAAT. These scripts can automatically extract the pitch from multiple sound files. It extracts the pitch by sampling the pitch at intervals and using a method called autocorrelation for determining the pitch at each moment in time, Jiménez-Hernández (2016) explains in depth how autocorrelation works. The script for extracting the pitch is on the GitHub page along with the rest of the code for the algorithm [2]. The pitch uses parameters to determine how to extract the pitch. These parameters are the time step, pitch floor, and pitch ceiling.

[2] https://github.com/danielvhvs/Bsc_Thesis_AI

The time step specifies the time between samples for the pitch. The pitch floor specifies the lower bound. Since there is always background sound we need to know below what value we do not want to sample pitch, pitch floor determines what is the boundary, pitch ceiling specifies the upper bound in the same way. In Fig. 2, there is an example of a sound file visualized in PRAAT. The blue line is an example of the pitch for certain parameter settings.

Region of interest The sound files do not start when a participant starts uttering the sentence or stop when someone stops speaking. Praat returns "–undefined–" when there is no pitch. The first step is to remove the values outside of the utterance. The second step is to take a range of the remaining sound. The full sound file contains many pitch values. The preprocessing takes a range for two reasons. First, limit the amount of information for the model. Second, sound files vary in length. Taking a region of interest ensures the length of time is always the same. Intuition tells us that the end and beginning are often the most important for pitch changes. Therefore, the method takes the first 0.5 and last 0.5 seconds and removes the rest. The 0.5 was an estimate. This research focuses more on the use of cues. Hence, the range was not varied.

Interpolation Some sentences can have gaps, like in the example in Fig. 2. The gaps result from PRAAT's pitch floor having a limit on the sound. Some words do not register a pitch, but there is sound. There is no silence. Interpolation helps in removing the gaps. Interpolation is a mathematical method for approximating unknown values based on the known surrounding values. Quadratic interpolation seemed to give the best transition for the gaps. Fig. 3(a) shows visible gaps in the pitch. Fig.3(b) shows the pitch without any gaps after interpolation.

Normalization Speech is not a smooth signal. Speech has many small changes over time, but these small changes do not matter, so the logarithm is taken from the data to normalize it. People have different pitch tones for their voices. Preferably, the pitch tone is more general. Hence, the starting pitch was taken and subtracted from the whole sentence for each sound file. Therefore, the start pitch is now 0 for every sentence. The start is normalized to 0 since assumed here is that how much the pitch changes with respect to the start is only important. An example sound after logarithmic scaling and normalization to 0 at the start is visible in Fig.3(b).

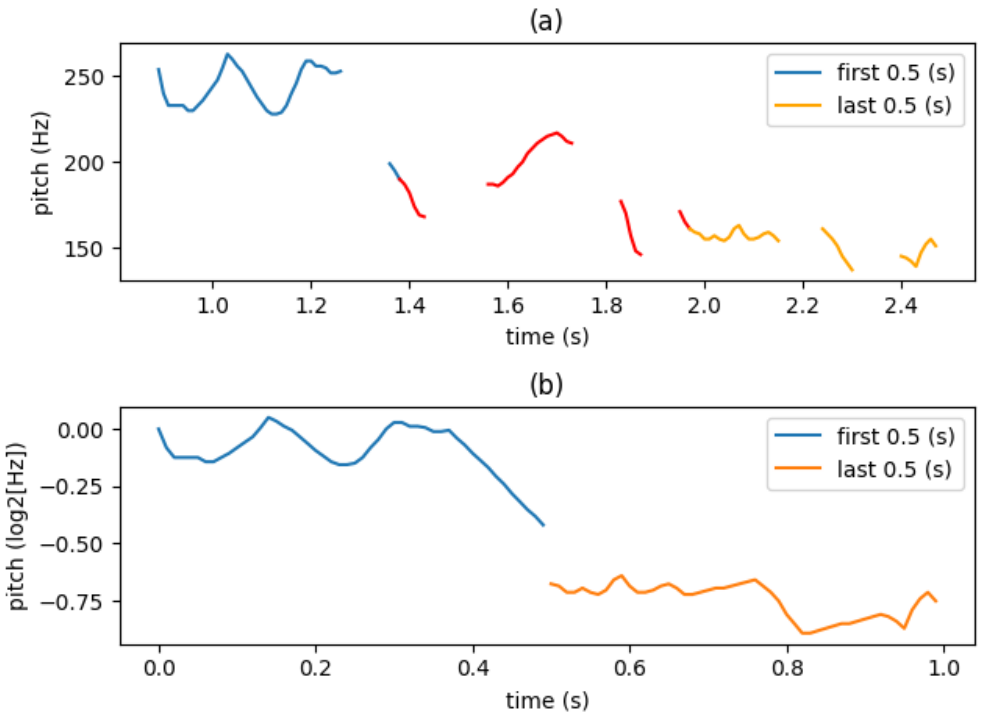

Figure 3: (a) shows the pitch of a single sound file over time before interpolation (b) shows the pitch after interpolation and normalisation and taking the range for the start and ending of the sound

Abstracting Pitch data has a lot of small slight changes. It is necessary only to find the important changes. The goal is to create an abstract representation of the pitch so that most of the noise is gone and only the important patterns are left. The assumption is that only change is important and not absolute values. There was a lot of trial and error in exploring the abstraction method. The exploration resulted in two methods for this research. There are multiple steps for abstraction. The start is the same for both. First, the pitch needs to be smoothed out. This step is trying to remove most of the small variations. Fig. 4 shows this in two graphs. Fig. 4(a) shows the original pitch from the sound file after the interpolation and normalization. Fig. 4(b) shows the data after smoothing.

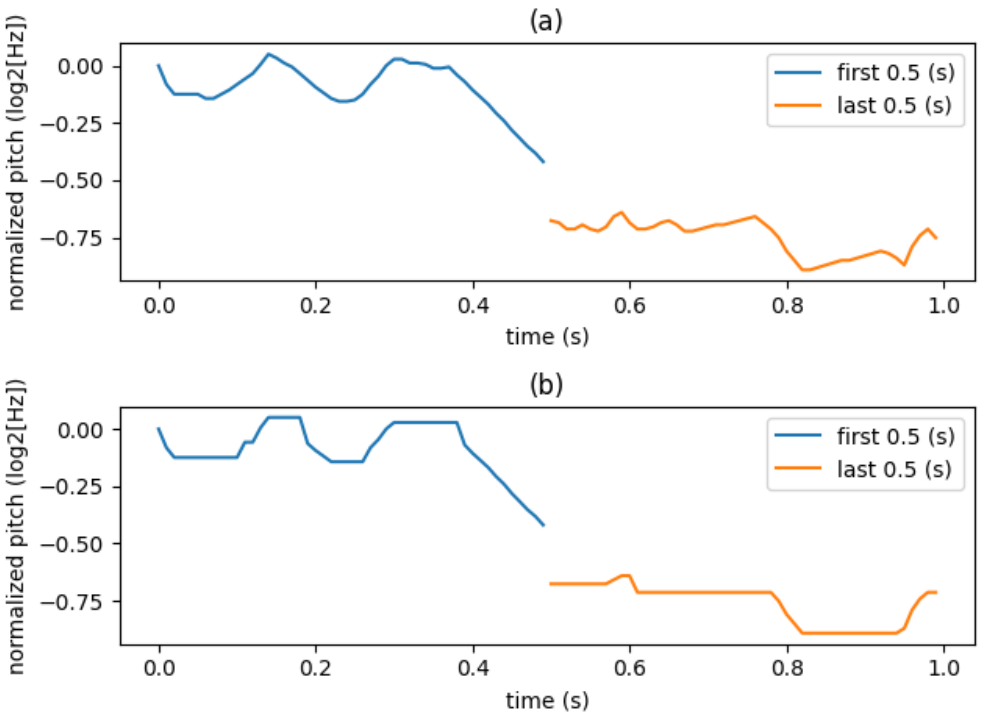

Figure 4: (a) shows the pitch of a single sound file over time (b) shows the smoothed out pitch from (a) using the smoothing algorithm described in the section for abstracting

Smoothing takes multiple steps. First, the program calculates the gradient for the data at each timestep. An algorithm goes through the single data sentence. The program starts at the first timestep, and the data value increases or decreases if the gradient exceeds a specified boundary value. If the gradient is too low, the data value stays constant at the value of the previous time step. Therefore, the data has fewer small variations because the small variations disappear. We can see where the data increases, decreases, or stays flat. For the first method, this is where the smoothing step is over. The second step is to turn the abstract representation into cues. The second step is to check where there is an increase or decrease with respect to the flat areas. In this process, there is a check for every flat area. If before a flat area, the gradient increases, the flat area is a "step up" and gets assigned the letter H for higher. Likewise, if before a flat area, the gradient decreases, the flat area is a "step down" and gets denoted as L for lower.

There is some extra processing for the second method of abstraction. Instead of just looking at a decrease or increase, there is an extra step after creating Fig. 4. All gradients are removed, resulting in Fig. 5(a). Only the flat areas remain. The remaining flat areas are important for the next step. The program searches for the first acceptable flat area starting from the first flat area. An accepted flat area is long enough, and the difference is high enough. Hyper-parameters determine the requirements for the minimum length and minimum pitch difference. For example, look at Fig. 5(a). The second flat area is only one dot, which might not be large enough if the minimum length is 3. The flat area after that is four which might be long enough. The second criterion needs enough difference, which might still not be enough. If the difference has to be at least a logarithmic base 2 value of 0.3, then there are no acceptable flat areas. Logarithmic base 2 is the scale used throughout the preprocessing.

After removing the unacceptable flat areas, the graph might look like 5(b) based on the parameters for minimum area length and difference. The program removed some flat areas there. After the program finds the first acceptable flat area, it searches for the next acceptable flat area w.r.t. the previous flat area.

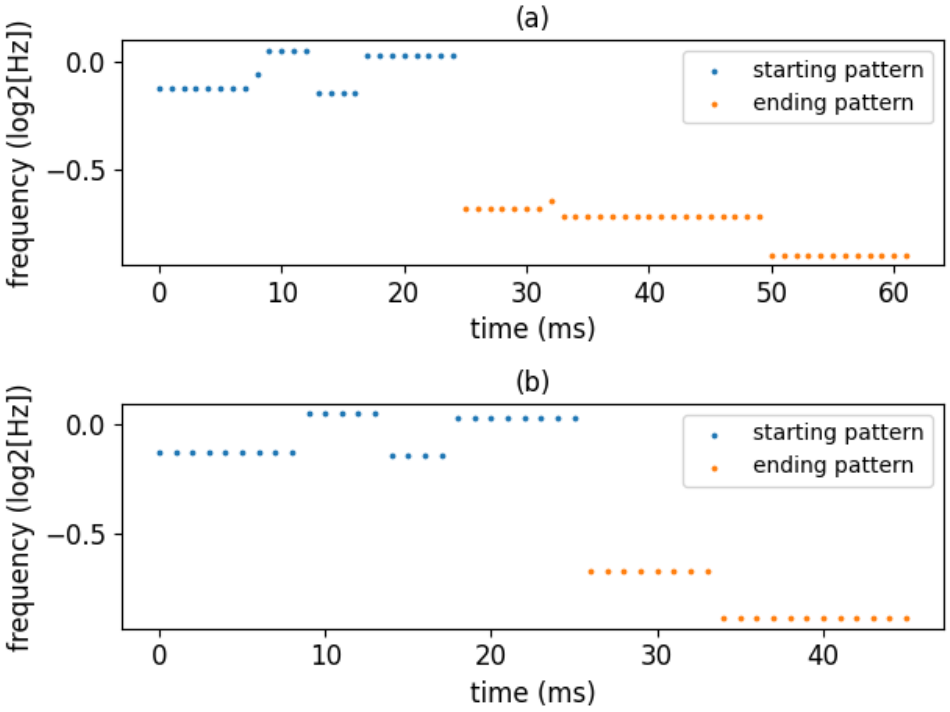

Figure 5: This figure shows an example of one sound data point where only the flat lines are kept after smoothing. These lines are transformed to cue strings in the form of L and H's. There is a start and ending pattern. The blue dots represent the starting pattern, which is #HLH. The yellow dots represent the ending pattern, which is #L. The patterns are created from a relative change. The starting pattern start with one flat area, the next flat area is higher, hence an H for higher. The third flat area is lower than the second, hence an L for lower. This continuous indefinitely resulting in the aforementioned cue strings.

The acceptable flat areas in Fig. 5(b) represent the abstract representation for method 2. The program can transform this abstraction into cues directly. From the first flat area, if the next one is higher, it gets an H for higher. It gets an L for lower if it is lower than the previous flat area.

For the first preprocessing step, the time range leaves the start and end of each sound file. This results in two lists of pitch data, one for the start and one for the end. Consequently, the algorithm keeps track of two lists through the whole abstracting progress, and the output is two strings with the letters H and L using either method 1 or method 2 for abstracting.

Cues For the error-driven learning algorithm, there need to be cues for training. The cues have to be determined from the abstracted input, which is two lists of the letters H and L for higher and lower. There can be many variations in the data, as mentioned before. Consequently, there can be many variations in the patterns of H and L. Multiple L's may follow each other, and vice versa, multiple H's could follow each other. Furthermore, there is no limit to the length of each sequence. It all depends on how many recognized variations are in the sound file. It can be a sequence of more than a 100 H and L's. An example input string can be "HLLH" and "LLHH" for one sentence. The order of the changes is important. The start and end need to be specified, using the # to mark the edges. # at the left marks the start, and # at the right marks the end. The strings are then "#HLLH" as the start and "LLHH#" as the end. The cue strings are from the full region of interest. For the model, cue sets are created using two hyper-parameters based on the cue strings. These hyper-parameters specify how much of the full cue the program uses. Fig. 6

shows a visualization of how the cue strings are mapped onto cue sets according to the length specified by the hyper-parameters. One parameter specifies the maximum length from the start, and one specifies the end. For example, specifying two for the start results in the following: "#H" and "#HL" as two different cues. Vice versa, specifying three for the end results in "H#", "HH#" and "LHH#" as three different cues. Specifying five for the example would return the same cues as four since the pattern only has a length of four. All possible inputs for the model are cues up to a length specified by the hyper-parameters like just mentioned. These cue sets map onto the cues for the error-driven learning model. Fig. 7 shows a visualization of the error-driven learning network with the cues connected to the weights. Some cues connect to the outcomes, but others do not connect to the outcomes. This depends on the cue string. The cue string #HLLH connects to #H but not too #L. Only the cues in the cue set have a connection to the outcome. If for one data sentence there is not a cue #L it might be for another sentence. Every type of cue connects to outcomes for at least one speech signal but not for all of them. Fig. 7 is just an example. These might connect differently for another speech signal.

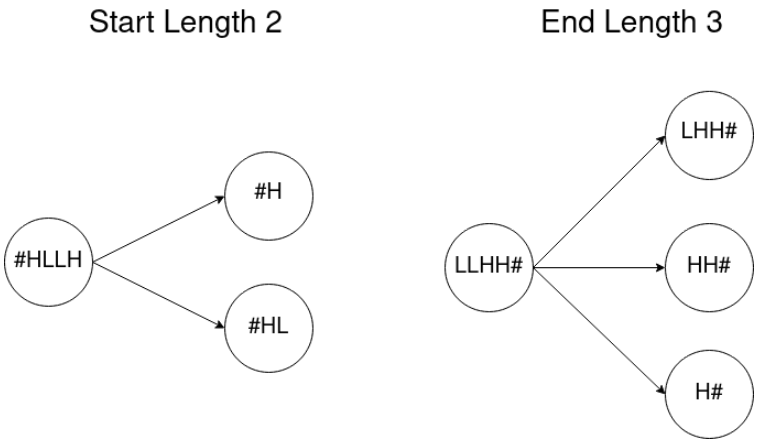

Figure 6: This is a visualisation of how certain cue strings map onto a certain set of cues. The left hand side of both mappings is the cue string after abstraction. The right hand side is the cues the string maps onto based on the hyper-parameter for maximum cue length indicated at the top. For the left mapping there is a maximum length of two. Hence, #HLLH will not map onto #HLL since that is a length of three. For the right mapping there is a maximum length of three. The maximum length for the start and end does not have to be the same length.

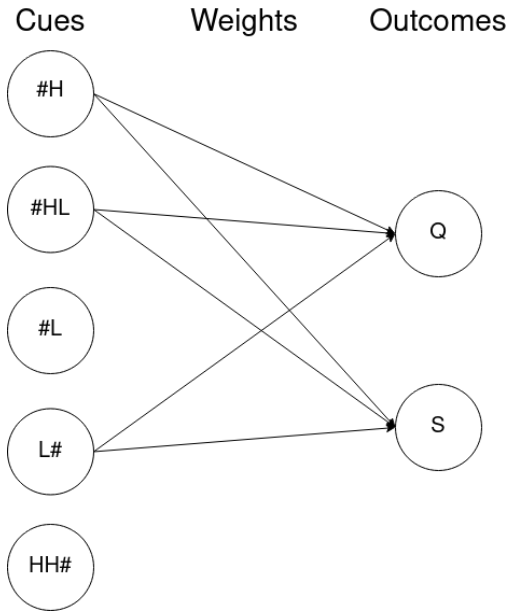

Figure 7: This is a visualisation of what an error-driven learning network looks like for this research. These are the type of cues used for this research which represent the changes in the pitch. This is an example for one data sentence. The cues that are not connected to the outcomes are not part of the cue set for that particular data sentence. In this case a sentence that has start, #HL and end L#.

2.3 Hyper parameter selection

Several hyper-parameters affect the preprocessing and the training process. First, there are the parameters for the pitch extraction in PRAAT, time step, pitch floor, and pitch ceiling. Second, there is a boundary for the gradient. Third is the extra parameters for the second abstraction method. These are the boundary for the minimum length of a flat area and the minimum difference between two subsequent flat areas. Fourth are the parameters for determining the cues for the training. These are the maximum length of the starting cues and the maximum length of the ending cues. Last are the hyper-parameters needed for the error-driven learning algorithm. The regular Delta rule is used so only η can be varied.

The parameters in PRAAT are set to 0, 75, and 500 for the time step, pitch floor and pitch ceiling respectively. These are the default values when using the graphical user interface for PRAAT. Hence, these are the chosen hyper-parameters for PRAAT. The boundary for the pitch smoothing process came from an initial exploration. Several values were plotted for several sound files. The visual interpretation guided the initial value for the boundary value. Certain important features in the data might be ignored if the boundary value is too high. However, if the boundary is too low, it may be too detailed, and even small insignificant features remain. Fig. 8 shows an example of a visual representation. This process provided a value of 2.4 as an initial value. 2.4 gave abstractions that seemed close enough to the real data without being too detailed. This process aided in closing in on the range for the hyper-parameter tuning, as there is no information on what is a good range. The hyper-parameter tuning tested values around this value.

The rest of the parameter tuning had two steps. The regular parameters like boundary, flat area and the rest. The second step is the maximum length of the cues, which is necessary for the sub-research question. This is analysed more in-dept. Hence, this is a second step.

All of the parameters were chosen based on the training and validation. No exploration was needed for the other parameters. The boundary value is the only parameter that required exploration as there is no

intuition for the value to use. Several runs were performed and the values that gave the highest estimate score was picked. For doing the runs cross-validation was used to limit variations based on the data or training process. For each different hyper-parameters combination there 40 different splits that are always the same. Hence, every combination of hyper-parameters was trained 40 times and tested 40 times.

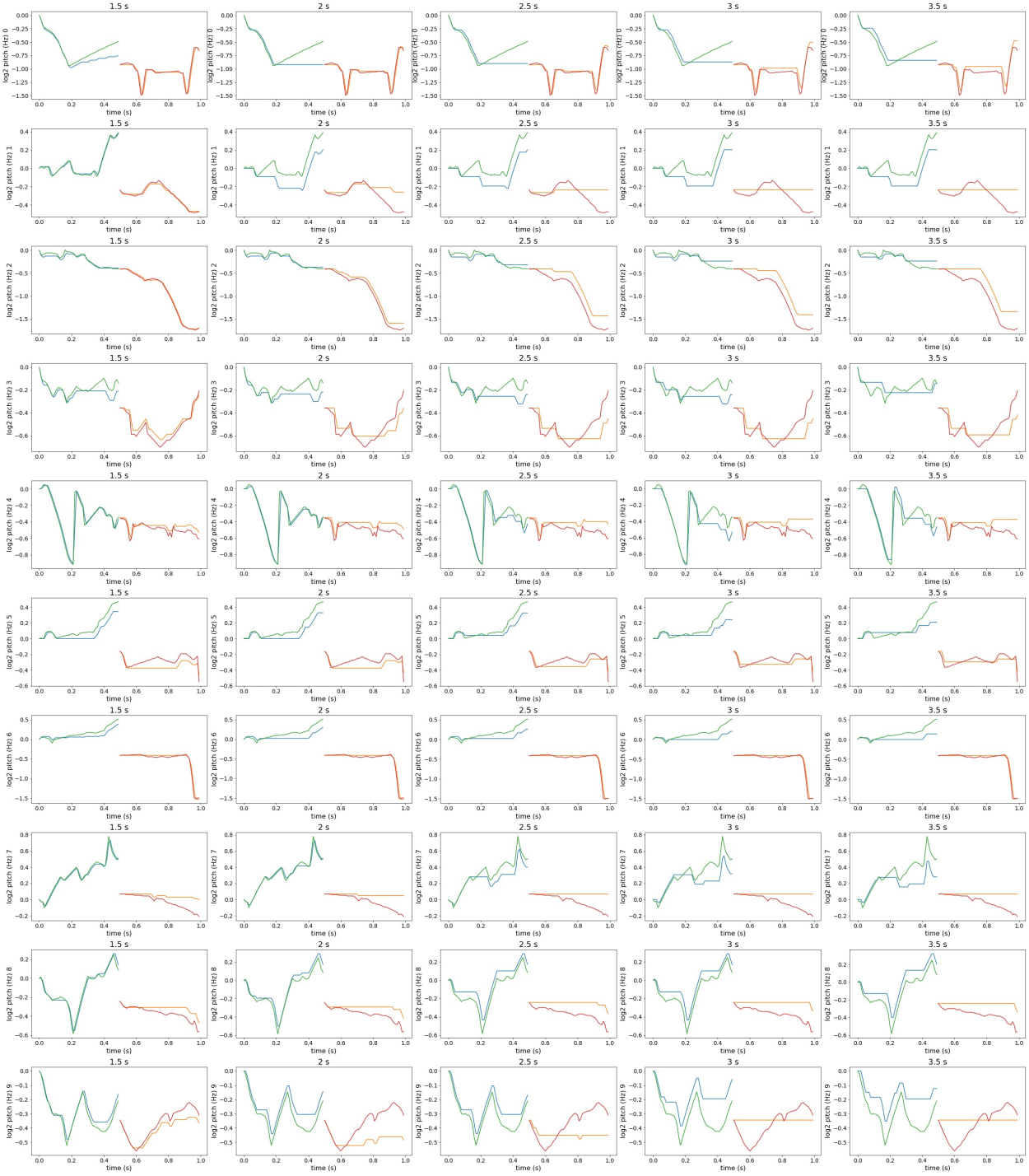

Figure 8: This shows an example of graphs for different levels of abstraction for multiple sound files for questions. The green and red line are original pitch for each sound file. The blue and orange line are after smoothing with the process from the methods. The title of each individual graph shows what parameter value was used for the boundary of the gradient. The letter is whether it is a statement (s) or a question (q). For this example there are no questions. The boundary value changes horizontally and the sound file changes vertically. Each different row is a different sound file.

2.4 Model training process

The first steps of the pipeline was creating a method that creates cues from the pitch. The second step is to do the actual training of the model and then evaluate the model on different hyper-parameters settings and analyse different cue sets. The main research question is to find out the possible accuracy that is attainable with this method of using error-driven learning on only pitch. Hence, it is necessary to test multiple hyper-parameters to find out the best possible set of hyper-parameters. This research also looks at a sub question. What is the most informative part of a sentence, that is why the extra analysis of the different cue sets is important.

validation settings for hyper-parameter tuning The training data is split into training and validation data to train the and see which hyper-parameters to pick to train the whole set and then test on the actual test data set. We split the data before preprocessing and training into 80% training data and 20% testing data. The optimal ratio is about 70/30 for five parameters in a linear regression model (Joseph, 2022). There is not much data available, so the data is split into 80/20 to use as much data for training. For cross-validation, the training data is again split into 80% for training and 20% for validation. There is a skew in the data towards the statements. There are fewer questions in the dataset. It is necessary to account for the bias in the data. Hence, the questions are over-sampled for the training data so that the amount of questions is the same as the statements. For the evaluation, the same is done to use balanced accuracy as a metric for comparison. After hyper-parameter tuning, the model uses the full 80% training data and tests on the 20% test data.

implementing error-driven learning The training uses error-driven learning, as mentioned before. The error-driven learning algorithm is implemented in R using the 'edl' package[3] (van Rij & Hoppe, 2021). This package helps to skip the tedious process of implementing the algorithm. The library implements a training process based on cues and outcomes. The training updates the weight for a specified number of iterations. The data is shuffled before the training starts. The cues represent H and L, as mentioned in the introduction and preprocessing sections. After the training is over, the model can evaluate the test data. Based on the weights and the cue sets for each data point, the algorithm gives a different activation for both outcomes.

predictions based on probability For making predictions, the model needs the probability that the algorithm assigns to some cue set belonging to some outcome. The output of error-driven learning is an activation value. The program uses two functions for calculating the probability. The first one is the RELU function for activation (Agarap, 2018). The RELU function sets negative values to 0, and all other values are linear, so they do not change. This is necessary as the Luce choice function cannot properly use negative values.

The second one is the Luce choice function (Luce, 1959). Luce's choice axiom formulates that the probability of picking one item from a group of items should not be affected by whether other items are present. This formulation translates into the formula for calculating the probability in Eq. 6. In this case, the function denoted by u is the RELU function.

$$P(a|A) = \frac{u(a)}{\sum_{a \in A} u(a)} \text{for some value function u : A} \rightarrow (0,\infty) \tag{6}$$

After that, these functions calculate the probability of a set of cues being either a question or a statement. The highest probability outcome is the "guess" the algorithm made. The amount of correct guesses evaluates the accuracy of the model. In the end, there is a model with optimized hyper-parameters. This model can

[3]https://cran.r-project.org/web/packages/edl/edl.pdf

make predictions with some weights by relating it to the activation and then to the probability as just mentioned.

Maximum cue length analysis The next step is determining the optimal hyper-parameters. As mentioned, this is a two-step process. Section 2.3 already describes the hyper-parameter tuning. The first step only requires hyper-parameters to be determined for the gradient boundary, minimum flat area length, and minimum flat area length. The second step is analyzing the optimal maximum cue length hyper-parameters for start and end. The analysis involves looking at how changing the maximum cue length affects the model. This part of the analysis is essential for answering the sub-research questions. The sub-research question was about which part of a sentence gives the most information to the model for making the predictions. After determining the optimal hyper-parameters, the analysis for maximum cue length can start. This research will look at how varying the maximum length affects the accuracy. Furthermore, by setting the maximum length for either start or end to 0, we can evaluate the effect of not using any starting or ending cues. After doing the analysis, we know the effects of the maximum cue length, and in addition, we know the optimal length, which we can use for performing the training and testing on the entire data set.

Testing set analysis The last step is to analyze the cue-outcome weights and the model's accuracy. As mentioned before, each cue connects to an outcome. This connection has a weight. After the previous steps, the right cues from length analysis and the optimal hyper-parameters are known. The model trains on the full train data set and makes predictions based on the testing data. The weights are analyzed after the training. The weight analysis is different from looking at the different lengths. Each cue is different. #L is different from #H. The weights tell us which cue is more important for predicting statements and which is more important for predicting questions. Analyzing the length and the weights tells us what information is most important for predicting sentences, and it answers the sub-research question. However, the accuracy and other metrics are evaluated on the test set to answer the main research question.

3 Results

As discussed in the methods, cross-validation is important for finding the optimal hyper-parameters. The hyper-parameters were picked based on histograms with balanced accuracy. Balanced accuracy is the accuracy for balanced validation data. Examples are in the appendix. The optimal hyper-parameters after the search are in Table 3 for both abstraction methods.

hyper-parameter	gradient boundary	minimum flat area length	minimum flat area difference
abstraction method 1	3.4	-	-
abstraction method 2	2.8	1	0.16

Table 3: Hyperparameters used for both abstraction methods

The length of the cues was checked more in-depth with the parameters in Table 3. The analysis of the cue length resulted in the histograms shown in Fig. 9 and Fig. 10. The numbers at the bottom of each bar show the maximum cue length for the starting cues (left) and the ending cues (right). In both methods, there is a decrease in accuracy when not using any ending cues. Secondly, for both methods, the accuracy is slightly decreased when not using any beginning cues. However, for the second method, this is more clearly visible.

The difference between the methods is that the first method increases in accuracy when using a longer maximum cue length up to a length of 4. The second method does not have any increase in accuracy. Furthermore, the second method has a higher accuracy than the first method. Method 2 has higher accuracy than method 1, so further analysis looks more in-depth for only method 2.

The second method has no increase in accuracy for longer cue length. Consequently, further analysis uses only a maximum length of one as a higher length does not improve the model, and it is important to keep the model simple.

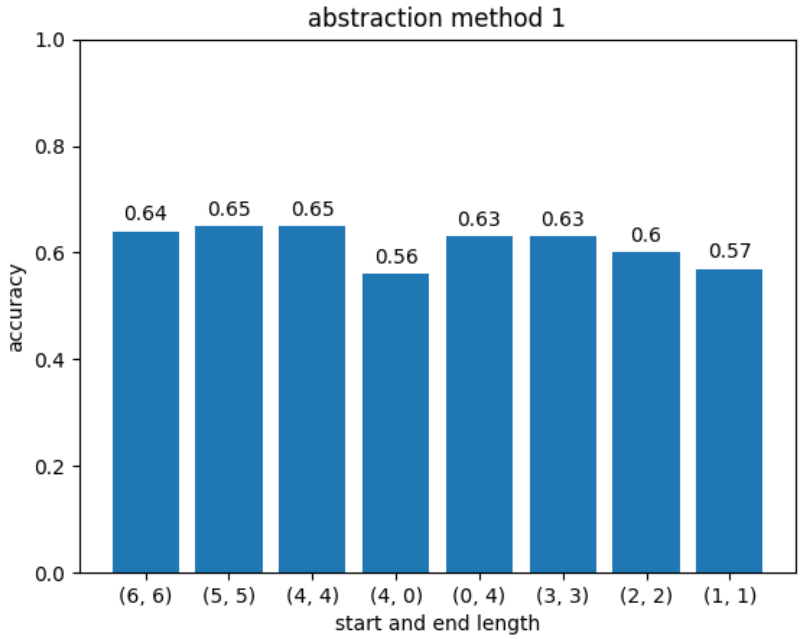

Figure 9: The accuracies for the different lengths for testing on the first abstraction method. The numbers at the bottom of each bar show the maximum cue length for the starting cues (left) and the ending cues (right)

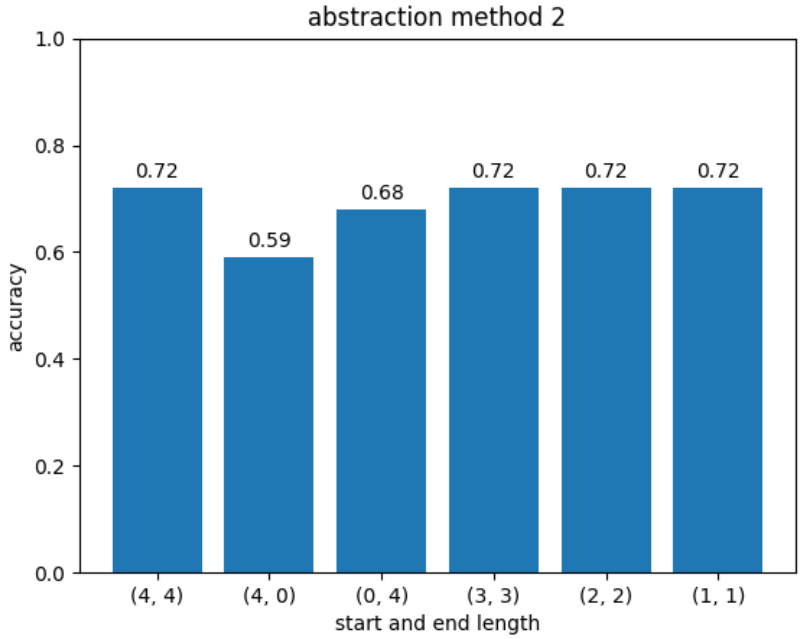

Figure 10: The accuracies for the different lengths for testing on the second abstraction method. The numbers at the bottom of each bar show the maximum cue length for the starting cues (left) and the ending cues (right)

After hyper-parameter fitting, training the model on the full training data was necessary, and then testing on the test set. This results in a list of predictions. The accuracy of these predictions is listed in Table 4. The balanced accuracy is the accuracy for a balanced test set. The training is always balanced to avoid bias. Only the balanced accuracy requires a balanced test set. Balancing means that the data set oversamples the outcome with fewer samples, so there are equal amounts of questions and statements. The F1-score is a useful metric for quantifying unbalanced data, as in this case. The F1-score combines recall and precision. It is high only when both recall and precision are high (Zhang & Zhang, 2009).

The F1-score is taken for each class and then averaged. Sometimes the weighted F1-score can be taken, but that shows a higher bias to the data set with more samples which is not what we want. The questions are much worse at being predicted than statements. All metrics are worse for questions than for statements. There are more questions predicted as statements than the other way around.

	recall	precision	F1-score
question	0.537	0.384	0.447
statement	0.760	0.855	0.804
balanced accuracy			0.649
averaged F1-score			0.626

Table 4: Table with cue weights and frequencies

The cues have different frequencies with which they occur in the training dataset. Table 5 shows a summary of the training data set statistics. It also shows the weights of the connections between cues and outcomes. The model is binary, meaning if the weight connection to an outcome is positive, then the connection to the other outcome is the same but negative. Fig. 11 visualizes the weights in a histogram. The higher the weight, the more predictive it is for a certain outcome. For example, L# is higher than #H. Hence, if both are present as a cue, which is possible since it is the end and beginning, then it is predicted as a statement since L# predicts a statement. The observation from the weights is that if there is any starting cue, they are predictive for questions, but for the ending, it is dependent. If the pitch increases at the end, then a sentence is a question. If the pitch decreases, then it is a statement. If there is no change in the pitch through the whole sentence, meaning there are no other cues, then it is the empty cue, which is a statement.

cue	count S	count Q	weight	weight category
empty	706	438	0.091	statement
L#	1934	1066	0.173	statement
H#	313	1143	0.161	question
#H	920	1472	0.125	question
#L	393	572	0.146	question

Table 5: Table with cue weights and frequencies for abstraction method 2

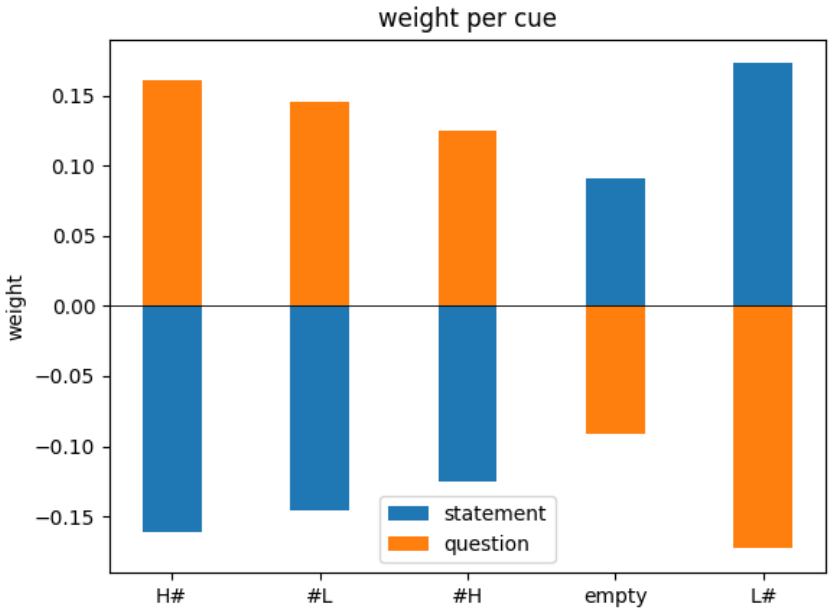

Figure 11: The weights for every cue connected to statement (blue) or question (orange) for the second abstraction method

There are some different visualizations of the distribution of the data that show the distribution of cues in Fig. 12, 13, and 14. The pie charts show the relative occurrences of each starting, ending cue and the empty cues for questions and statements. Fig. 12 shows the starting cues. The #H cue is more frequent for questions than for statements, which is the same for the #L cue. Starting and ending cues can co-occur, so the ending cues are in a different chart. The empty cues are in a different chart since the empty cue is both an ending and starting cue. The pie chart shows that some cues are more frequent for questions than statements. When comparing these pie charts to the weights in Fig. 11, thre is an observation to make. If the frequency of cues is higher for one outcome than for the other, then the weights are also higher. The empty cue has a higher frequency for statements than for questions, similarly the weight is positive for statements.

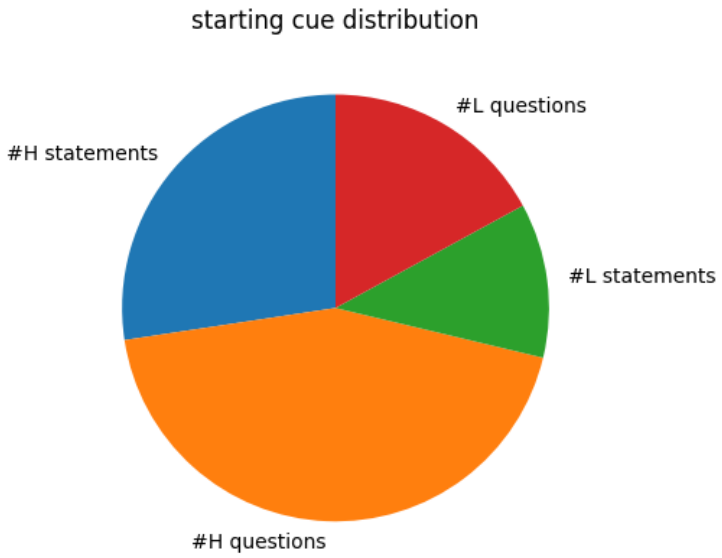

Figure 12: A pie chart of the number of occurrences of each starting cue for both statements and questions

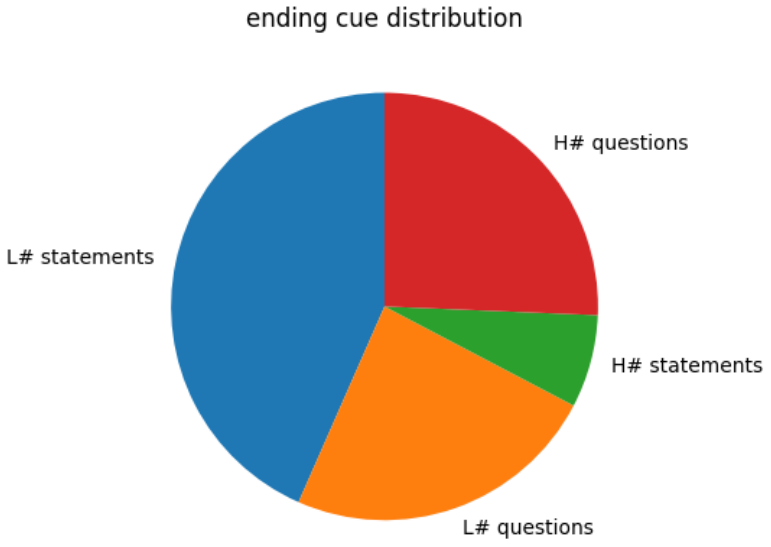

Figure 13: A pie chart of the number of occurrences of each ending cue for both statements and questions

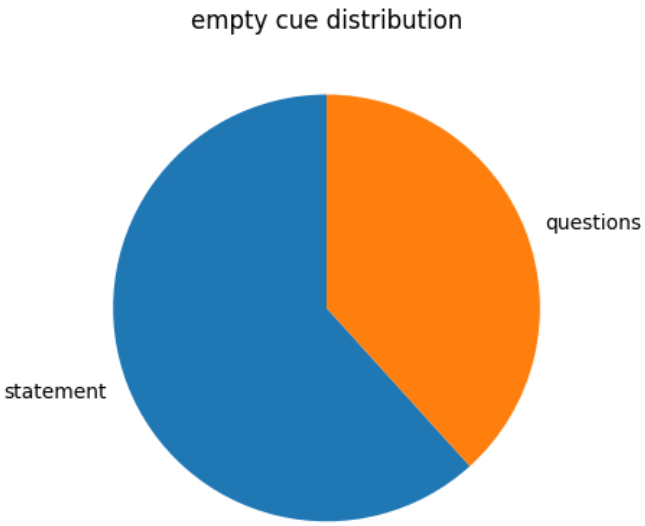

Figure 14: A pie chart of the number of occurrences of the ending cue for both statements and questions

4 Discussion

From the results, multiple observations can be made. First, comparing the cross-validation for both the first and second abstraction methods shows an observed increase in the balanced accuracy for the second abstraction method. The second abstraction method apparently removes extra noise the first method did not remove. Secondly, observing different lengths for the cues in the sound files shows us insights into how the model uses the cues. The insights are then that the model does not need cues that are longer than one change in the intonation. The model only needs to use a small amount of information. The other insight is that the intonation at the start and end both give information about the classification. However, the ending of a sentence is more significant in helping make correct predictions.

The model trained with the optimal parameters and the optimal cues. The chosen method was then the second abstraction method with the hyper-parameters in Table 3 with a length for both the starting and ending cues of one. The model then gave a set of weights between cues and outcomes, which are the actual results together with the predictive metrics like F1-score and accuracy. The metrics show some problems with the model. The recall, precision, and F1-score for questions are much lower. There is a bias for statement sentences. There are more predictions for questions as statements than the other way around.

4.1 Limitations and assumptions

There are some limitations to the model and some necessary assumptions. The metrics for questions and statements show a result from the first limitation. The recall, precision, and F1-score for questions are much lower. If there is a question, then it is more often predicted as a statement. The training data was balanced to account for bias in the distribution of statements and questions, but the problem remains. Usually, statements have no change in pitch, which is apparent from the empty cues being more frequent for statements. Pitch differentiates the questions by changing the pitch from statements. That means it is easier to recognize statements since they have less variation in pitch than questions. There is much noise in terms of changes in the intonation when speaking. The noise complicates the process when the goal is to recognize changes in the pitch for making the classifications. Different speakers have varying individual differences in their pitch increases and decreases. What is an adequate difference for one speaker might be different for another speaker. It is not easy to generalize between different speakers. In addition, the second limitation is the amount of data used. When analyzing any language that is not English, even for a widely spoken language like Spanish, there is less data available. Furthermore, the data must be full sentences, with transcriptions to differentiate statements from questions for supervised learning. There is not much data that meets the requirement. The low amount of data makes it more difficult to generalize and eliminate the small variations because of noise and speaker differences.

There were also some assumptions. First, the model only uses part of the sentences, only 0.5 seconds at the start and end of each sentence. The assumption was that the only important information was at the start and end of each sentence. Furthermore, to do this, the assumption is that every person speaks with equal speed. Otherwise, it makes no sense since the change might occur in different places. Of course, people do speak at different speeds.

There were some other assumptions about the preprocessing. In the data, there are gaps. These gaps are consequences of how PRAAT analyses data with the floor and ceiling boundary for the pitch. The assumption is that every speaker has the same pitch range. This assumption is an approximation to avoid finding the specific pitch range for each speaker.

Lastly, the model only used pitch as a feature. In the real world, people use context to determine what someone says. The context makes it easier to hear questions when someone has a smaller difference in the change of intonation. This lack of context limits the highest accuracy obtainable. It will only be possible to

gain 100% accuracy with all the necessary information.

4.2 Improvements and further research

The limitations already account for some possible improvements. Furthermore, there are problems for possible future research. The limitations already talked about individual differences in speakers. We have to use a default setting for all speakers or do it individually for every sound file and visually inspect the data. Better would be to make this process automatic. Instead of humans picking the parameters, train a machine learning model to get a sound file as input and then output a list of parameters to get a specific pitch output for the individual speaker. This process could automate the preprocessing to make it more accurate. However, automatic machine learning like this is difficult. The other improvement mentioned before as a limitation was the amount of data. The amount of data is always difficult and a common problem with languages. More data is necessary to improve the generalizability.

Furthermore, what about further research? The accuracy is not a 100% or even close. The accuracy is above random chance which is good. We only looked at the pitch as a feature basically for making predictions. The next step is to make a model that does use context as a feature. If it still cannot get a 100% accuracy for making predictions, then something is wrong.

Second, the model might use different pitch patterns than human native speakers of the Argentinian Spanish dialect. It is possible to make a dataset of artificial data of all gibberish sentences. The sentences do not have any meaning to Spanish speakers, so they cannot use any context for the meaning of the words. The important information comes from artificially changing the pitch in a similar pattern to the model's predictions. If the human native speakers have the same predictions for whether they are questions or statements as the model, then we can say these patterns model the real world well.

5 Conclusions

This research aimed to explore prosody in Spanish. More specifically, to see how well a model can predict whether a sentence is a statement or a question using only prosody. Error-driven learning was chosen as it is interpretable and efficient for cases with less data. The challenge was that error-driven learning uses cues instead of continuous values. As the pitch value is continuous, it was necessary to transform the data to feed it to the model. No methods were available for the transformation, so developing a process that made an abstract representation from pitch to use as cues as part of the pipeline was necessary. The full pipeline created was the developed abstraction process and the training process.

From the start, several limitations were evident. There is little data available. Second, there is much noise in the data. There were several assumptions to account for these limitations. Furthermore, using only the pitch limits the predictive power as people use more than just intonation. Despite the limitations, a model could eventually be trained. The cues used represented increasing or decreasing pitch intervals in the data. Focusing on relative changes is useful since it is much less dependent on the speaker. From the cross-validation after training, the data taught us that using the full cue set was unnecessary. It was only necessary to use the first and last pitch interval change in each sentence for making predictions. It is still necessary to use both the start and end of a sentence for making these predictions, but the ending is better for making accurate predictions. The trained model showed that the model could make predictions for the testing data above random chance. The predictions gave an unbalanced accuracy of 64.9% and an averaged F1-score of 62.6%.

The reason for the accuracy not being 100% is attributed to the fact that the model does not use context clues like meaning and previous spoken sentences by the speaker like humans do in the real world. However, we can make accurate predictions despite using only intonation for the model, and the answer to the research question is that our predictions are above random chance at 64.9% accuracy. Further research is necessary to evaluate the model's accuracy by using either extra features or testing on native speakers.

6 Appendix

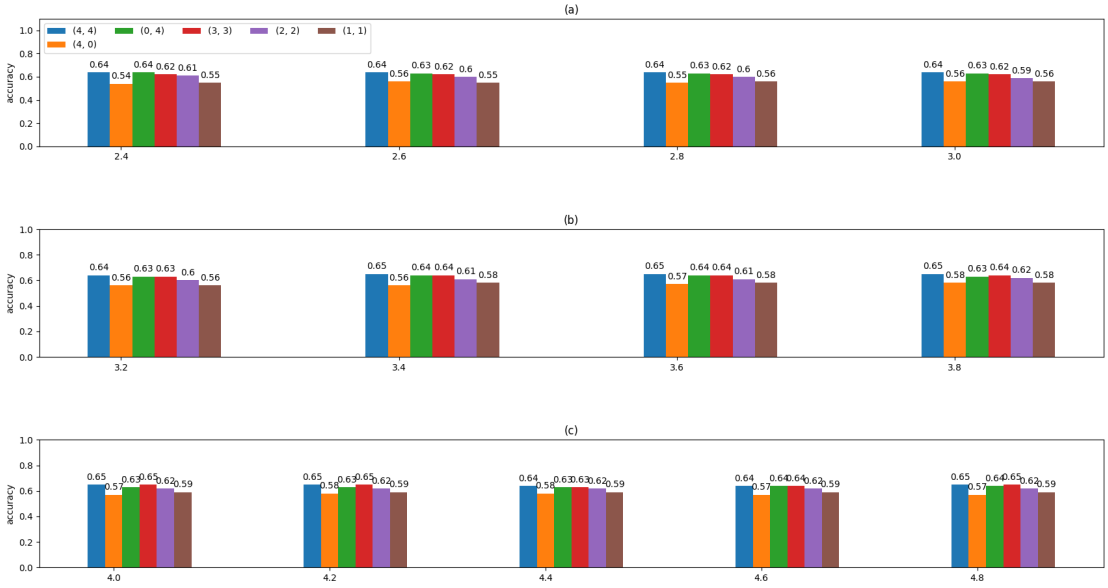

Figure 15: The accuracies for the hyper-parameters for testing on the first abstraction method

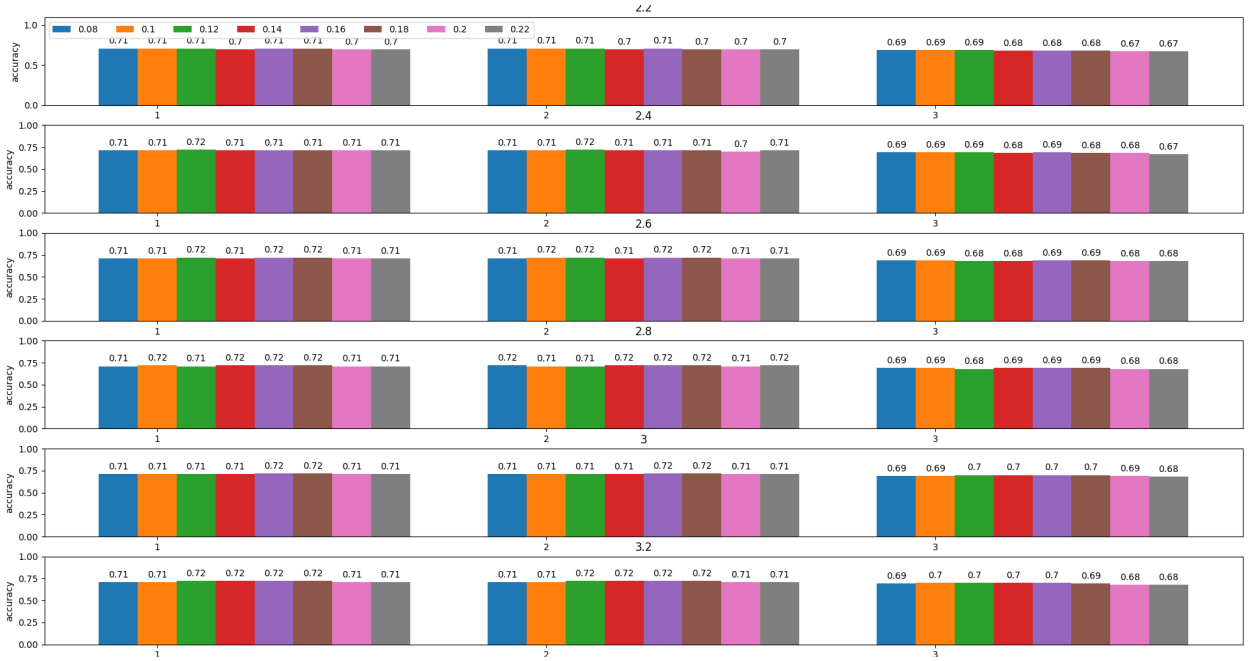

Figure 16: The accuracies for the hyper-parameters for testing on the second abstraction method

References

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Agrawal, S., Shruti, A., & Krishna, C. R. (2010). Prosodic feature based text dependent speaker recognition using machine learning algorithms. *International Journal of Engineering Science and Technology*, *2*(10), 5150–5157.

Arnold, D., Tomaschek, F., Sering, K., Lopez, F., & Baayen, R. H. (2017). Words from spontaneous conversational speech can be recognized with human-like accuracy by an error-driven learning algorithm that discriminates between meanings straight from smart acoustic features, bypassing the phoneme as recognition unit. *PloS one*, *12*(4), e0174623.

Guevara-Rukoz, A., Demirsahin, I., He, F., Chu, S.-H. C., Sarin, S., Pipatsrisawat, K., ... Kjartansson, O. (2020, May). Crowdsourcing Latin American Spanish for Low-Resource Text-to-Speech. In *Proceedings of the 12th language resources and evaluation conference (lrec)* (pp. 6504–6513). Marseille, France: European Language Resources Association (ELRA). Retrieved from `https://www.aclweb.org/anthology/2020.lrec-1.801`

Hoppe, D. B., Hendriks, P., Ramscar, M., & van Rij, J. (2022). An exploration of error-driven learning in simple two-layer networks from a discriminative learning perspective. *Behavior Research Methods*, *54*(5), 2221–2251.

Jiménez-Hernández, M. (2016). A tutorial to extract the pitch in speech signals using autocorrelation.

Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *15*(4), 531–538.

Kamin, L. (1969). Predictability, surprise, attention, and conditioning. in ba campbell & rm church (eds.), punishment and aversive behavior (pp. 279-296). *New York: Appleton-Century-Crofts*.

Litman, D., Hirschberg, J., & Swerts, M. (2000). Predicting automatic speech recognition performance using prosodic cues. In *1st meeting of the north american chapter of the association for computational linguistics*.

Luce, R. D. (1959). *Individual choice behavior: A theoretical analysis, new york, ny: John willey and sons.* Inc.

McIntosh, C. (2013). *prosody.* Cambridge University Press.

Napier, R. M., Macrae, M., & Kehoe, E. J. (1992). Rapid reacquisition in conditioning of the rabbit's nictitating membrane response. *Journal of Experimental Psychology: Animal Behavior Processes*, *18*(2), 182.

Panda, S. K., Jena, A. K., Panda, M. R., & Panda, S. (2023). Speech emotion recognition using multimodal feature fusion with machine learning approach. *Multimedia Tools and Applications*, 1–19.

Paul, B., Bera, S., Dey, T., & Phadikar, S. (2023). Machine learning approach of speech emotions recognition using feature fusion technique. *Multimedia Tools and Applications*, 1–26.

Rescorla, R. A. (1972). A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and non-reinforcement. *Classical conditioning, Current research and theory*, *2*, 64–69.

Shagi, G., & Aji, S. (2022). A machine learning approach for gender identification using statistical features of pitch in speeches. *Applied Acoustics*, *185*, 108392.

van Rij, J., & Hoppe, D. (2021). *edl: Toolbox for error-driven learning simulations with two-layer networks.* (R package version 1.1)

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... others (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhang, E., & Zhang, Y. (2009). F-measure. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 1147–1147). Boston, MA: Springer US. Retrieved from `https://doi.org/10.1007/978-0-387-39940-9_483` doi: 10.1007/978-0-387-39940-9_483