Universidad del País Vasco Euskal Herriko Unibertsitatea

Contributions to Natural Language Processing and Hierarchical Classification

by

César Montenegro Portillo

Supervised by Roberto Santana and Jose A. Lozano

Donostia - San Sebastián, November 2023

To my family
and friends

Acknowledgments

First of all, I would like to thank my advisors Roberto Santana and Jose Antonio Lozano for their patience and guidance during my Ph.D journey. My thesis would not have been completed without their great support and encouragement.

I am also very grateful for my friends in the Intelligent Systems Group with whom I shared these years, and also to the members of other research groups at the Faculty of Computer Science. Their camaraderie and intellectual exchange during lunch or seminaries have enriched my experience and contributed significantly to this work.

Last but not least, I would like to thank the endless support of my wife, family and friends, which have always been my support and greatest treasure. This thesis is also for them.

Contents

1 Preliminaries

1.1 Context

Population ageing is one of the pressing social challenges of the 21st century, affecting not only Europe but also other developed communities. The decline in fertility rates and the increasing life expectancy have resulted in a rapid increase in the proportion of individuals aged 65 and above. In Europe, it is projected that this percentage will rise from 16% in 2010 to 29% in 2060 [67]. Furthermore, this ageing population is expected to be predominantly female, with a growing number of individuals over 80 years old. Within this context, numerous social challenges need to be addressed:

- Meeting the high demand for infrastructures and services that cater to the needs and preferences of the elderly, including addressing the desire for aging at home while considering cultural factors and the diversity of societies.
- Adopting a cross-cutting approach that spans various sectors such as health, housing, support for low-income individuals, and ensuring suitable living conditions in both urban and rural areas.
- Supporting healthy older individuals in maintaining productivity and independent lives within their communities for as long as possible. This entails the development of a wide range of products and services that facilitate a certain standard of living and well-being as people age, with technology playing a crucial role in many innovative solutions.

Despite advances in healthcare and technology, most elder care is currently provided by informal caregivers, such as friends and family members. However, predictions suggest that this type of care will decrease in the future. Therefore, studies encourage society to focus on improving the lifestyles of the elderly, helping them remain independent for more extended periods.

The EMPATHIC (Empathic, Expressive, Advanced Virtual Coach to Improve Independent Healthy-Life-Years of the Elderly) project[1] has significantly advanced technology in this field. The focus of this project is on researching, innovating, and validating new interaction paradigms and platforms for future generations of personalized Virtual Coaches (VC) designed to promote active aging. Central to this initiative is the development of the EMPATHIC-VC, an unobtrusive, emotionally expressive virtual coach. Its primary goal is to engage senior users in adopting healthier lifestyles, encompassing aspects of diet, physical activity, and social interactions. By actively reducing the risk of potential chronic diseases, the VC helps seniors maintain an independent and enjoyable life, which, in turn, assists their caregivers. The main objective of a VC is to establish a connection between one's physical state and emotional well-being. To achieve this, it employs multi-modal face, eye gaze, and speech analytics modules to perceive and identify users' social and emotional states. Additionally, it learns and comprehends users' requirements and expectations, responding adaptively to their needs through innovative spoken dialog systems and intelligent computational models. This combination of modules facilitates real-time interaction between users and the coach, promoting empathy and enhancing the user experience.

The EMPATHIC project employs coaching concepts through a VC, implementing coaching strategies designed to induce behavioral changes in users. Coaching dialogs are structured logically, following a question-answer model, with the aim of comprehending the user's needs, limitations, and objectives (Montenegro et al. [61], Justo et al. [40]). It's important that the user accepts these goals, so the coaching dialogs promote self-awareness and offer guidance toward setting realistic and healthy objectives. Each dialog adheres to a sequential structure based on the GROW framework (Goal, Reality, Options, and Will of an action plan), a framework developed by health coaching professionals.

A general overview of the EMPATHIC-VC workflow is depicted in Figure 1.1. The EMPATHIC-VC takes audio and video inputs, which are then processed by various modules. The Dialog Manager (DM) is a crucial component that orchestrates the coaching session, determining the system's actions during each agent's turn.

Regarding Natural Language Processing (NLP), three modules work in sequence to provide the DM with a precise description of the user's dialog turn in terms of meaning. These modules are the Automatic Speech Recognition module (ASR-M), End-of-turn Detection (EOTD-M), and the Natural Language Understanding module (NLU-M). The ASR-M transcribes speech into text as a continuous stream and sends it to the EOTD-M, which identifies when the user has finished their turn and is awaiting a response from the EMPATHIC-VC. Once the turn has concluded, all transcribed words are

[1] http://www.empathic-project.eu/

forwarded to the NLU-M, responsible for analyzing the meaning of the user's speech and providing processed information to the DM.

Moving closer to the user interaction, once the DM has determined how to proceed with the coaching session, the Natural Language Generator generates a response, taking into account both the form and content of the message. This response also considers additional input sources, such as information extracted from facial expressions, voice tone, or external resources like weather forecasts and cultural events. The text-based response is then converted into audible speech by the Text-to-Speech module and delivered to the User Interface, ensuring effective communication with the user.

Figure 1.1 illustrates the most important modules in the virtual coach, and the flow of information as designed.

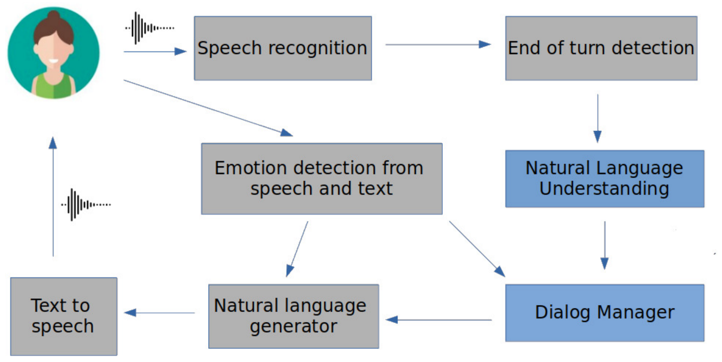

Speech recognition
End of turn detection
Emotion detection from speech and text
Natural Language Understanding
Text to speech
Natural language generator
Dialog Manager

Fig. 1.1: EMPATHIC architecture modules related to the dialog act taxonomy definition.

1.2 Overview of the dissertation

In this dissertation, we present four contributions made during the development of the EMPATHIC project. These contributions are divided into two distinct areas: contributions to NLP and contributions to Hierarchical Classification.

The dissertation's first part starts with an introduction to the EMPATHIC-VC modules within the NLP domain (Chapter 2). The first contribution of this dissertation introduces an Automatic Speech Recognition simulator (ASR-SIM) that implements various strategies for the End-Of-Turn detection task (Chapter 3), specifically designed for scenarios with error-prone input. This ASR-SIM proves especially valuable for evaluating the impact of ASR-M errors on NLU-M and can aid in training more robust NLU-M systems.

The second contribution in the first part of the dissertation proposes a dialog-act taxonomy based on the EMPATHIC project's requirements (Chapter 4). This taxonomy elucidates two of the NLU-M tasks, namely, Intent classification and Topic classification.

The taxonomy introduces two Hierarchical Classification (HC) tasks, serving as the foundation for the subsequent contributions in the field of HC presented in the second part of this dissertation (Chapter 5). These contributions extend beyond the scope of the EMPATHIC project and offer potential applications in diverse domains.

The primary focus of the first contribution is to address labeling deficiencies that may arise within HC problems (Chapter 6), resulting in the formulation of Weakly Supervised Hierarchical Classification problems (WSHC). We demonstrate the advantages of incorporating hierarchical information during the training process, leading to significant improvements in both performance and computing time. We conduct a comprehensive comparative analysis of a designated strategy across three different WSHC scenarios, carefully examining the implications of integrating hierarchical characteristics into the learning process.

The second contribution introduces the Multi-Dimensional Hierarchical Classification paradigm (MDHC) (Chapter 7), inspired by the Intent and Topic classification tasks of the NLU presented in the dialog-act taxonomy in the first part of the dissertation. Alongside presenting this paradigm, we propose specific methods, performance measures, and a procedure for generating synthetic MDHC scenarios to facilitate future studies and evaluations in this area.

The dissertation concludes with a chapter dedicated to summarizing the general findings and outlining potential avenues for future research (Chapter 8).

Part I

Contributions on Natural Language Processing

2

Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and natural languages [44]. The term natural language is used to distinguish human languages (such as English, Spanish or French) from computer languages (such as C++, Java or Python). The techniques developed within this sub-field of artificial intelligence aim to provide computers the ability to understand commands given in natural language and perform according to it.

The ASR-M, EOTD-M and NLU-M present in the EMPATHIC-VC are framed within the NLP field. Figure 2.1 illustrates the architecture of each of the modules. In the ASR-M architecture, the feature extraction component takes as input the raw audio signal, filters noises that do not correspond to human speech frequencies, and extracts frequency-domain feature vectors that are used to feed the following acoustic model. The acoustic model, leveraging knowledge of acoustics and phonetics, estimates one or multiple sets of words that best align with the given feature vectors and assigns them corresponding scores. The language model evaluates the correlation between words learned from a training corpora and estimates scores for each hypothesized sentence. The hypothesis search component then combines the scores from both models for each hypothesis, ultimately producing the recognized sentence as the word sequence with the highest score [125].

The audio signal received by the ASR-M is a continuous stream of audio, and as a result, the ASR-M outputs a stream of words with timing information, which could be hundreds of words long in a whole conversation. A conversation between two humans consists of a turn-taking transference of information, and replacing one of the humans with a bot requires the detection of the user's End-Of-Turn pauses. The goal of an EOTD-M is to detect this change of turn in a conversation between a human and the system. This triggers the evaluation of the sentence or sentences received by the NLU-M. The main task of the NLU-M is to convert these sentences of human language into more formal representations, making them easier for computer programs to interpret. In this case, four characteristics of the sentences are evaluated:

- **Topic:** Identifying the main subject or theme of the sentence.
- **Intent:** Determining the purpose or goal behind the sentence.
- **Polarity:** Assessing whether the sentence conveys a positive, negative, or neutral sentiment.
- **Entities:** Recognizing specific named entities or objects mentioned in the sentence.

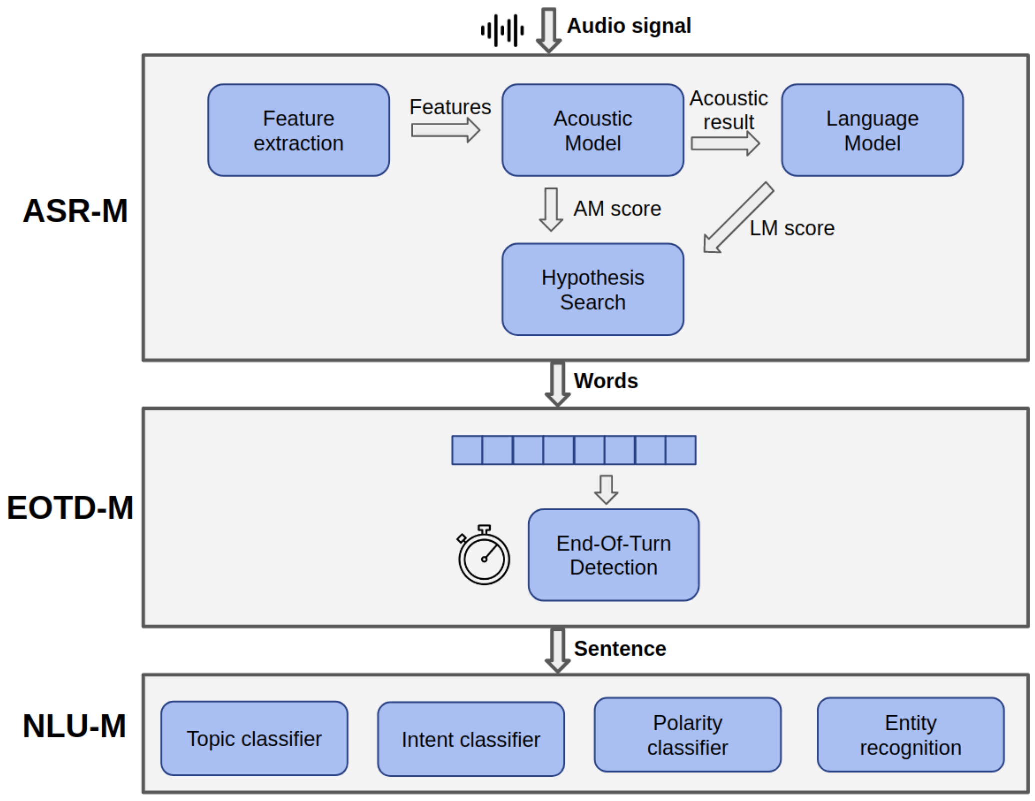


Fig. 2.1: ASR-M, EOTD-M, and NLU-M schematic architectures.

As a result of the development of the EOTD-M and NLU-M, in this section we present two contributions. The first contribution studies how word errors in ASR-M can affect the performance of the EOTD-M. The second contribution presents the taxonomy defined for the Topic and Intent characteristics detected in the NLU-M, which also affects the design of the DM.

3

End-of-Turn detection task

3.1 Introduction

The audio signal received by the ASR-M is a continuous stream of audio. The system must filter the human voice from ambient noise, and estimate the best group of words that corresponds to the audio signal. As a result, the ASR-M outputs a stream of words with timing information, which could be hundreds of words long in a whole conversation. A conversation between two humans consists of a turn-taking transference of information, and replacing one of the humans with a bot requires the detection of the user's End-Of-Turn pauses. The goal of an EOTD-M is to detect this change of turn in a conversation between a human and the system. This triggers the evaluation of the sentence or sentences received by the NLU-M.

The consequences of failing on the EOTD task are:

1. **Anticipation:** When the NLU-M receives an incomplete sentence, the system may potentially answer while the user is still talking, causing overlap between the speech of the human and the system. Some systems close the users microphone [18] when answering, missing all the information transmitted by the user during the overlap.
2. **Excessive delay:** when an End-Of-Turn is not detected in time, the time gap between a real End-Of-Turn and the reply from the system is too high, and the user experience is harmed by unnatural waiting times between turns.

Several aspects have to be considered when designing an EOTD-M. Particularly relevant are the architecture of the spoken dialog (which defines the input to the EOTD-M) and the features used in the detection problem.

The architecture of a Spoken Dialog System can limit the input resources of an EOTD-M. Figures 3.1a and 3.1b illustrate how two common architectures differently condition the input of the EOTD-M. In Figure 3.1a, the EOTD-M receives information exclusively from the ASR-M, while in Figure 3.1b not only can ASR-M information be received, but also raw audio data. We can find studies in the literature that are based on the architecture of Figure 3.1a, such as the work by [81], who study the impact of the prediction power of features extracted from pause, prosodic, timing, lexical, syntactic and semantic information. Nevertheless, it is more common to find studies using features extracted from raw audio data, following the architecture in Figure 3.1b. There are different features that can be extracted from raw audio data, [18] extracted 40-dimensional log-Mel filterbanks with an upper limit of 4kHz and a frame step of 10ms using a 25ms window, while [55] and [3] used raw pitch (F0), smoothed F0 contour, Root Mean Square signal energy, the logarithmized signal energy, intensity, loudness, MFCC and smoothed pitch.

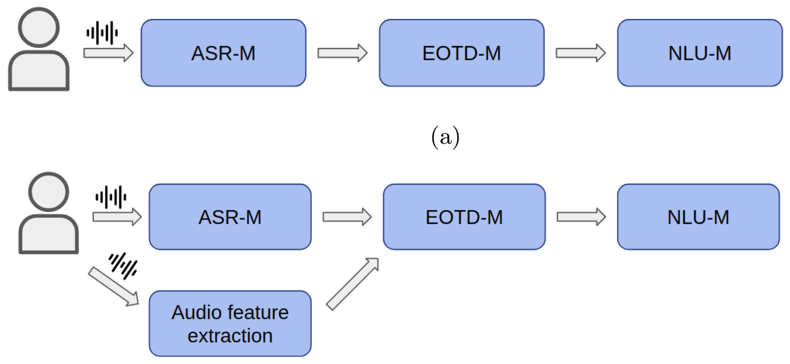


Fig. 3.1: Subfigure (a) shows an architecture where the EOTD-M uses the output of the ASR-M as input. Subfigure (b) shows an architecture where the EOTD-M uses the output of the ASR-M as input, but also has access to other features extracted from raw audio.

These two architectures exploit only the user's speech information, but different architectures can offer more sources of information, for example the architecture presented by [57] uses the user's utterance in conjunction with the interlocutor's utterance.

3.2 Sources of errors in ASR-M

One of the most challenging aspects of ASR-M is the mismatch between the training and testing conditions, or real life acoustic conditions. During testing, a system may encounter new recording conditions, microphone types, speakers, accents and different sources of background noise. Furthermore, even if the test scenarios are seen during training, there can be significant variability in their statistics [94]. Without specific noise-robust processing, even state-of-the-art speech recognition degrades rapidly under decreasing Signal-to-Noise Ratios [66].

These conditions will produce the following errors in the ASR-M transcription result:

1. **Confused word (substitutions):** Due to the pronunciation, noise, or even the accent, some words can be mistranslated. This often occurs when two words are phonetically similar.
2. **Missing word (deletion):** Sometimes due to noise, accent or other speech particularities, word sounds can be confused with ambient noise or unintelligible sounds.
3. **Extra word (insertion):** Although some ambient sounds can be confused with words, the most common source of word insertion occurs when the phoneme of a word can be represented by a tuple of words, instead of the true corresponding word. For example the tuple of words *"Join in"* could replace the word *"Joining"* because they are phonetically similar.

The Word Error Rate (WER)[126] defined below (Equation 3.1):

$$WER = \frac{S + D + I}{N} \tag{3.1}$$

where S, D, I and N are the number of substitutions, deletions, insertions and number of words in the reference respectively, is a common metric used to measure the performance of an ASR-M or machine translation system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level, and it is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors [64].

3.2.1 Speech profiles

The problems exposed above are related with the conversion of sound waves to phonemes, but there are other characteristics that are useful for communication and are related to the timing and duration of other language resources. These characteristics are: pronunciation speed, speaking rate, and pause duration.

Each person has their own way of speaking. And not even a combination of pronunciation speed, speaking rate, pause length or accent is fixed for a single person, it also varies depending on their mood or fatigue. Henceforth we will refer to the measurable set of these characteristics as **speech profile**. In subsequent sections, we introduce a speech profile representation and propose a way to obtain realistic values of the speech profile representation parameters from the analysis of real ASR-M outputs. For example, in Figure 3.2, the average letter duration of multiple speakers is compared, calculated as $letter\_duration = word\_pronunciation\_time/word\_length$. The figure shows the average letter duration grouped by word length. The data is extracted from the Switchboard dataset [25], which has become the de facto standard experimental testbed for speech recognition, and will be explained in more detail in Section 3.4. In Figure 3.2 it is possible to observe the profiles of the speakers that have the maximum and minimum average letter duration, as well as the profile of another 20 randomly chosen speakers. Figure 3.2 reveals that the fastest profile is double the speed of the slowest, illustrating how wide the range of speeds can be in a group of speakers.

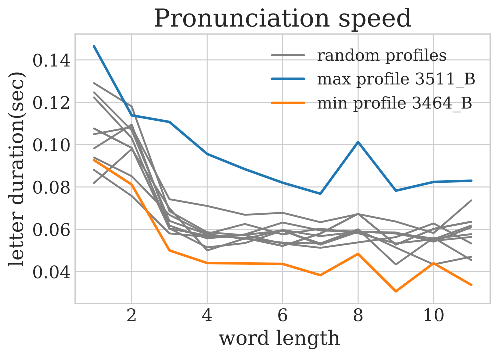


Fig. 3.2: Pronunciation speed profiles

3.3 ASR Simulator

As it is not possible to generate all possible types of noise that an ASR-M can receive, our goal is to introduce an ASR-SIM that can be controlled in such a way that the transcribed data exhibits different types and rates of artifacts. A characteristic feature of our simulator is that, instead of using an audio file as input, a dialog transcription or a plain text can be used. The ASR-SIM converts any conversation transcription into an ASR-M output with the desired probabilities of ASR-M errors, and desired speech profiles.

The ASR-SIM output format is composed of two differentiated parts:

1. Word information: Contains the possible words that the ASR-SIM may estimate that correspond to the audio fragment, and their confidence value.
2. Timing information: Indicates the timestamp of the start of the pronunciation of the word, and its duration.

The transformation from plain text to word and timing information will be determined by a number of internal parameters of the simulator. These parameters can be grouped into two classes: WER probabilities and speech profile parameters.

3.3.1 WER probabilities

The ASR-SIM allows us to set the probability of each particular error that makes up the WER (probability of a confused word, probability of a missing word, probability of an extra word). Given a sentence, for each word the three error probabilities are evaluated to determine if the word is affected by one of the defined errors. These errors were introduced in Section 3.2, and the description of the methods implemented to simulate each type of error follows:

1. Confused word: the word is substituted by a phonetically similar word. The phonemes of the replacement and replaced words will have a Levenshtein distance smaller than a given threshold. The timing information is calculated using the information of the substituted word.
2. Missing word: the word is substituted by a token that represents an unknown phoneme $< Unk >$. The timing information is calculated using the information of the substituted word.
3. Extra word: the phoneme of the word is randomly split into two subphonemes, and each one is replaced by a word with a phoneme with a Levenshtein distance smaller than a given threshold. The timing information is calculated using the original word information, proportionally sharing the word duration between the two replacement words, and with a pause $p = 0$ between replacements.

The implementation of the different error methods should mimic the errors of the ASR-M we want to simulate with the ASR-SIM. Even if an ASR-M uses a common set of features, the combination of the *Acoustic model*, the *Language model* and the *Hypothesis Search* make each ASR-M unique, and therefore the characteristics of the errors generated are unique as well. For example, for an ASR-M that gives more importance to the *Acoustic model* than to the *Language model*, when making a *confused word error* the true word might be replaced by a phonetically similar word, even if it causes an unlikely semantic error. In this example implementation the phonemes are obtained using the Refined Soundex algorithm, originated with the implementation of phonetic algorithms included with the Apache Commons library [24]. We use the Levenshtein distance [47] to compare word phonemes, as it is commonly used to compute error rates of ASR-M. We have used the implementation in the Pyphonetics library[1] for these tools, and the English dictionary from the

[1] https://pypi.org/project/pyphonetics/

Nltk library[2] has been used as a source of replacement words for the *Confused* and *Extra word errors.*

3.3.2 Speech profile parametrization

The parametrized characteristics of the speech profile used by the simulator are:

1. Word duration
2. Pause duration

While in theory these parameters could be arbitrarily set, a realistic output of the ASR-SIM will require a more sensible setting of the parameters. We address this issue using a statistical analysis of real ASR-M outputs. As described in Section 3.2.1, a speech profile can be defined by a set of characteristics. All these characteristics are measurable, and we can therefore generate a set of variables to simulate a particular speech profile, or simulate multiple speech profiles by modifying these variables. In order to perform the experiments that will be defined in Section 3.4, we will parametrize word duration performing the study described in Section 3.3.3, and pause duration (Section 3.3.4) based on [16]. These parameters will directly affect the codification of the sentences, since some codification methods use pauses between words as input information, and pause duration will affect the amount of evaluation points used in the experiments, as detailed in Section 3.4.1.

3.3.3 Word duration

In order to simulate word duration, we will use the values obtained by calculating the average and standard deviation of the letter duration of the speakers from the Switchboard dataset (Figure 3.2). The distribution of the values obtained for the letter duration calculus is illustrated in Figure 3.3. The ASR-SIM will use this empirical distribution to estimate the duration of each letter in a word, randomly sampling from the distribution. Although more sophisticated methods could have been used to estimate the pronunciation time of a particular word, this method maintains simplicity, and it still makes words last a proportional, but not fixed, amount of time for their length.

3.3.4 Duration of pauses

[16] present a large-scale study of silent pause duration, based on the analysis of read and spontaneous speech. Although in their study, spontaneous speech analysis is only performed in French, it does not represent an obstacle for our analysis since it has been observed that the language differences with respect

[2] https://www.nltk.org/

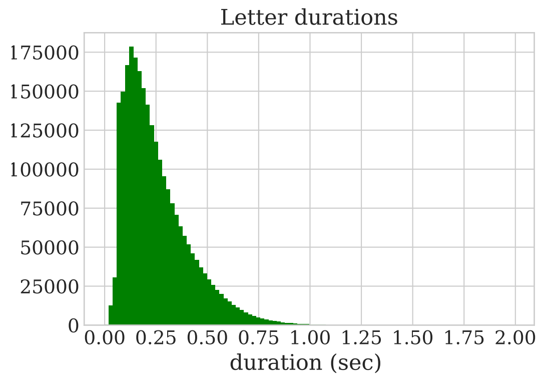


Fig. 3.3: Letter duration (ms)

to gap duration seem to be minor [118]. [16] made the hypothesis that the observed pause duration distributions are the result of a combination of three categories of pauses. By using *Generalized Reduced Gradient* (GRG2), they obtained a parametrized probabilistic model of the duration of pauses, which is described by Equation 3.2:

$$D(x) = k_1 N(\mu_1, \sigma_1, x) + k_2 N(\mu_2, \sigma_2, x) + k_3 N(\mu_3, \sigma_3, x) \tag{3.2}$$

where $D(x)$ is the distribution of the duration of pauses, $N(\mu_i, \sigma_i, x)$ is the normal law of mean μ_i, and their standard deviation is σ_i (duration of pauses are log-transformed). The parameters k_1, k_2 and k_3 represent the weight of each component distribution ($k_1 + k_2 + k_3 = 1$).

Based on this work, we match each of these pause duration distributions in increasing order of μ_i, with the pause between words, comma pause and dot pause respectively. The μ and σ values used for each distribution are shown in Table 3.1. Although μ values are available in the original study, σ values were deduced from the figures in [16].

The ASR-SIM will generate pauses according to these distributions when using plain text inputs. However, if the original pause information is provided, this information will be used.

Table 3.1: Pause distributions parameters.

	i	μ_i	σ_i
Between words pause	1	78	1.3
Comma pause	2	426	1.6
Dot pause	3	1585	1.3

Algorithm 1 Pseudocode of the ASR-SIM main function

```
*asr_output* = []
for token in source_text do
    if is_word(token) then
        is_confused = random() > confused_word_threshold
        is_missing = random() > missing_word_threshold
        is_extra = random() > extra_word_threshold
        possible_errors = [ ]
        if is_confused then possible_errors.append("confused")
        end if
        if is_missing then possible_errors.append("missing")
        end if
        if is_extra then possible_errors.append("extra")
        end if
        error = random_selection(possible_errors)
        if error == "confused" then
            confused_word = get_close_match(token)
            asr_output.append(confused_word)
        else if error == "missing" then
            asr_output.append(<unk>)
        else if error == "extra" then
            word1, word2 = generate_extra_word(token)
            asr_output.append(word1)
            asr_output.append(pause_between_words)
            asr_output.append(word2)
        else
            asr_output.append(token)
        end if
    else
        if token == "," then
            asr_output.append(comma_pause)
        else if token == "." then
            asr_output.append(point_pause)
        else if token == " " then
            asr_output.append(pause_between_words)
        end if
    end if
end for
return *asr_output*
```

Algorithm 2 Pseudocode of the generate_extra_word(word) function

```
1: w1_Length = random(1, length(word))
2: w2_Length = length(word) - w1_Length
3: w1_segment = word[:w1_Length]
4: w2_segment = word[w1_Length:]
5: firstWord = get_close_match(w1_segment)
6: secondWord = get_close_match(w2_segment)
7: return firstWord, secondWord
```

3.3.5 ASR Simulator pseudocode

The general procedure to transform a text to the desired ASR-SIM output is the one described by Algorithm 1. The output of the ASR-SIM contains the recognized words, and the associated timing information. Any character that is not a letter is removed from the input text, except for commas and periods which represent pauses in the speech.

In Algorithm 1, the source text is analyzed token by token (line 2). Whenever the token corresponds to a word (line 3), for each error type a random number is generated and compared to the associated threshold value (lines 8-10), the error to apply will be randomly selected between those that exceed the corresponding threshold (line 11), and the error mechanism is executed (lines 12-21). On the other hand, if the token is not a word, it will generate a pause based on the type of token (lines 25-30).

In lines 17-21, the extra word error is generated using an auxiliary function called generate_extra_word, which is in charge of generating two words from one, as explained in Section 3.2 and described in Algorithm 2. The generate_extra_word function randomly splits the word (lines 1-4) and finds a similar word for each segment with the function get_close_match (lines 5-6). This function is also used in line 10 to generate the confused word error by calculating the Levenshtein distance between the phonemes as explained in Section 3.3.

The source code is available under GNU General Public License v3.0 [3] in it's source code repository[4].

3.4 Experiments

In this section we present the experimental framework we have designed to evaluate the influence of the different types of errors in the behavior of the EOTD-M. We conduct this evaluation using the ASR-SIM presented. The parameters of the simulator are changed to produce different combinations of errors. The performance of the EOTD-M prediction task is then tested on

[3] https://choosealicense.com/licenses/gpl-3.0/
[4] https://github.com/CesarMontenegro/AsrSimulator

this simulated data. The section is organized as follows: Firstly, we describe the EOTD-M prediction task and the features used as input for the classifier. Then we describe the characteristics of the classifier, and the metrics used to evaluate the results. The two sections that follow describe the corpora used and how errors are generated by the ASR-SIM. Finally, we present and discuss the results of the experiments.

3.4.1 EOTD-M classification and sets of features

To evaluate the sensitivity of the classifier, we rely on the *Prediction at Pauses* task described by [100]. This is a standard turn-taking decision task that takes place at brief pauses during an interaction. The goal is to predict whether the user holding the turn is going to continue speaking (HOLD), or swap turns (SHIFT).

As mentioned in Section 3.1, there are multiple features that can be extracted from raw audio data, but since we use the ASR-SIM, our features will be those that can be extracted from transcriptions. Therefore, in this experiment, we compare the sensitivity of the classifier to the use of the following sets of feature vectors:

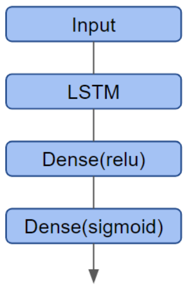
- Word Embeddings: multi-dimensional meaning representations of a word. For these experiments, we use the GloVe (Global Vectors for Word Representation [73]) pretrained embeddings[5]. This is a popular vector representation for Natural Language Processing tasks.
- POS Tags: part-of-speech tag for each word is considered to be a good predictor of turn-switches in the literature [30]. To obtain the tags, we use the tagger from the Nltk library, and generate a one-hot representation.
- Pauses: duration of time gaps between every pair of consecutive words in the sentence.
- Combined: a combination of Word Embeddings, POS Tags and Pauses.

3.4.2 Characteristics of the classifier

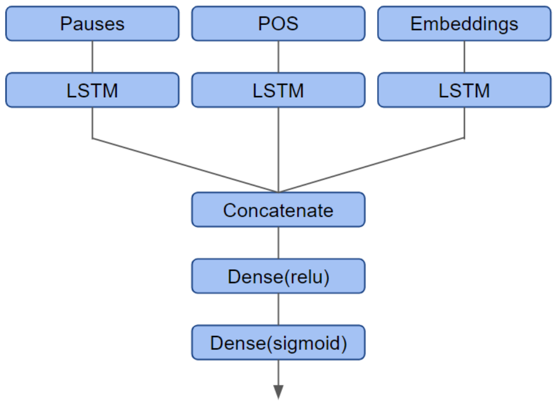
In the literature, the most frequently used models for EOTD-M are models based on LSTM Recurrent Networks [55, 3, 84, 50, 96, 56, 34]. Despite being combined with other layers or algorithms (*e.g.*, [96] add Convolutional Layers to the architecture), the main differences between them are the features they use to train the algorithm. Therefore, for our experiments, we will use the LSTM architectures illustrated in Figure 3.4, with the parameters described in Table 3.2.

For model validation, each scenario generated in these experiments will divide the dataset into three subsets. These three divisions will be called *Train*, *Validation* and *Test*. The algorithm will learn from the Train subset,

[5] Available online at https://nlp.stanford.edu/projects/glove/



(a) Single-feature input architecture



(b) Multiple-feature input architecture

Fig. 3.4: Subfigure (a) shows the architecture of the models that use one of the three features vectors as input. Subfigure (b) shows the architecture of the models that use the three types of feature vectors as input.

Table 3.2: Architecture of the models used, *Layer(type)* is the name of the units used in each particular layer and *Units* is the number of units in the layer.

Layer (type)	Units
LSTM	128
Dense(relu)	128
Dense(sigmoid)	1

while Validation is used to avoid overfitting by stopping the training process when the validation loss stops improving. The parametrization of this anti-overfitting mechanism is described by the *earlyStopping* variables in Table 3.3. The *patience* parameter allows the anti-overfitting mechanism to prevent the training procedure from stopping when it is temporally stuck in a local minima. Nevertheless, for these experiments we have observed that even low values of patience are enough to avoid local minima and overfitting.

3.4.3 Metrics

The LSTM network will output the probability of a sentence being a SHIFT pause. Using a threshold value, this output can be binarized, thus allowing to calculate the accuracy and other metrics. However, the determination of this threshold can severely affect the result. To avoid this drawback, in these

Table 3.3: Parameter of the training procedure for the LSTM based model, *n_batch* samples per gradient update, *epochs* is the number of epochs to train the model, *learning rate* controls how much to modify the model's parameters in response to the estimated error after each epoch, *pad_sequence* is the maximum length in number of words of a sentence (if the sentence is larger than this value, the first words in the sentence are discarded until it reaches the specified length), *earlyStopping monitor* is the variable monitored that will trigger the early stopping of the training procedure, *earlyStopping patience* is the number of epochs with no improvement after which training will be stopped, *loss function* is the optimization score function and *optimizer* is the name of the algorithm used to fit the parameters.

Parameter	Value #
n_batch	1000
epochs	200
learning rate	0.01
pad_sequence length	30
earlyStopping monitor	val_loss
earlyStopping patience	5
loss function	binary_crossentropy
optimizer	adam

experiments the Area Under the ROC Curve (AUC) will be used to evaluate the performance of the LSTM models.

3.4.4 Dialog data corpora

The experiments will be performed with two datasets. The first dataset will be based on the Switchboard dataset, for which we will not generate timing information (word duration and pause duration) since it already has that information available, and we will only induce the correspondent artifacts. This dataset is a telephone-speech corpus that consists of approximately 260 hours of speech and was originally collected by Texas Instruments in 1990-1991, under DARPA sponsorship[6]. It is a collection of about 2,400 two-sided telephone conversations among 543 speakers (241 female, 302 male) from all areas of the United States. In these types of conversations, where there is a lack of non-verbal communication, backchannel communication is very present. For End-Of-Turn detection, we are not interested in backchannel turns, we focus on the speech of the speaker who leads the turn, and is making a statement. The backchannel communication made by the listener on a turn is ignored, resulting in a dataset of 35,323 sentences.

[6] https://catalog.ldc.upenn.edu/LDC97S62

The second dataset will be based on the OpenSubtitles en-es corpus[7], for which the ASR-SIM will have to estimate all the timing information based on the parameters we have defined. The OpenSubtitles en-es corpus contains 61.4 million speech turns from movie scripts. These speech turns do not belong to real speech, the dialogs are scripted, and therefore the structure and vocabulary vary from natural human-to-human speech. Nevertheless, this dataset provides a complementary validation benchmark for our study, since each communication scenario presents a particular problem, such as telephone conversations, face-to-face conversations or videoconferences. We will train and test the EOTD-M on these types of dialogs without trying to export the models from the scripted-dialog environment to human-to-human speech. For the experiments, we will use 50.000 randomly selected sentences from this dataset.

According to the EOTD-M problem described in Section 3.4.1, each speech turn generates a SHIFT-labeled instance, while HOLD instances are generated from turns containing pauses longer than the specified threshold (δ = 1045ms). This is proposed in [16] as a cut between the distributions that we have associated with comma and dot pauses. A HOLD instance is the subsequence of tokens that precedes each pause greater than the threshold in a turn. Illustrative examples of the generation of SHIFT and HOLD instances can be found in Tables 3.4 and 3.5, where, from a hypothetical transcribed sentence, we analyze the pause duration to generate HOLD and SHIFT instances. The example in Table 3.4 uses a threshold value of δ = 1045ms, and the one in Table 3.5 uses δ = 1500ms.

Table 3.4: Example of the generation of SHIFT and HOLD instances with δ = 1045ms

Hypothetical sentence	
Hello, I would like to buy a necklace, a gold necklace.	
Pause duration	
Hello<1200ms>I would like to buy a necklace<1600ms>a gold necklace	
Instances and labels generated from the sentence (δ = 1045ms):	
Hello I would like to buy a necklace a gold necklace	SHIFT
Hello I would like to buy a necklace	HOLD
Hello	HOLD

Finally, the datasets are balanced to contain the same amount of SHIFT and HOLD instances. This is done by randomly sampling from each class.

3.4.5 Word Error probabilities

The probabilities of the ASR-SIM errors used to analyze the impact on the quality of the LSTM estimator are: $\{0.0, 0.1, 0.3, 0.5, 0.7\}$ for the three types of

[7] [dataset] http://opus.nlpl.eu/OpenSubtitles.php

Table 3.5: Example of the generation of SHIFT and HOLD instances with $\delta = 1500$ms

Hypothetical sentence	
Hello, I would like to buy a necklace, a gold necklace.	
Pause duration	
Hello<1200ms>I would like to buy a necklace<1600ms>a gold necklace	
Instances and labels generated from the sentence (δ = 1500ms):	
Hello I would like to buy a necklace a gold necklace	SHIFT
Hello I would like to buy a necklace	HOLD

errors. In order to identify which factors influence each feature representation technique, each error probability is analyzed independently. The threshold for the Levenshtein distance of *Confused* and *Extra word errors* is set to $\tau = 3$. To evaluate the impact of the different error types and probabilities, the experiments will be conducted following two strategies: *same_distribution* and *different_distribution*. The *same_distribution* strategy will apply the same error probability in train, validation and test sets, given a particular error type and probability. The *different_distribution* strategy will apply the error to the test set only, while the algorithm will train with free-from-error data. This second strategy simulates the scenario in which the training data is generated in a controlled environment, with a low probability of errors. However, the evaluation data is generated in a real environment, exposed to the errors defined in Section 3.2. Therefore, the *different_distribution* strategy will allow us to investigate how important training with the expected test error rates is.

3.5 Results

Before analyzing how the ASR-SIM transcription errors affect the performance of the LSTM based EOTD-M, we have measured how each error type affects each featurization technique. Inspired on the analysis performed by [113], where they investigate the effects of word substitution errors (*Confused word error*) on sentence embeddings, we have measured how much featurized sentences change under the effect of the different errors generated in the ASR-SIM.

3.5.1 Effects of the ASR-SIM errors on the featurization techniques

The errors considered in this work can cause variations in the length of a sentence, unlike in [113], where only the confusion error is analyzed, which does not change the number of words in a sentence. *Extra word error* adds words to the sentences, and this makes the approach of [113] unsuitable for this work, since it compares the original and modified sentences word to word.

Therefore, to overcome this difference, we treat the sentences as time series of feature vectors, and use the Dynamic Time Warping (DTW) measure [51] to compare sentences without errors and sentences with the induced errors. DTW is a measure that finds the optimal alignment between two time series by stretching or shrinking one of the time series along its time axis [87]. This warping between two time series can then be used to determine the similarity between the two time series by means of a defined distance measure. DTW is often used in speech recognition to determine if two waveforms represent the same spoken phrase [1].

DTW has been previously used to compare similarity between sentences [51], and although it does not guarantee the triangle inequality, it provides an estimation of how the errors affect the vectorized representation of the sentences. For this evaluation we have used the Python *FastDTW* library[8] based on the work of [87], which is an approximate DTW algorithm that provides optimal or near-optimal alignments with an $O(\mathrm{N})$ time and memory complexity.

We have randomly selected 1000 sentences to calculate the average distance to their modified version for each dataset, the distance is calculated by the FastDTW algorithm with the Euclidean distance between each pair of matched words. The distances have been calculated under the effect of the different error probabilities described in Section 3.4.5. This comparison exercise has been performed 10 times to take into consideration the variability generated in the ASR-SIM, and averaged to illustrate the results in Figures 3.5 and 3.6. These figures are composed of three plots each, one for each featurization technique. Each plot contains information on how a particular featurization technique is affected by the three error types with different error probabilities. The *Y axis* represents the average distance between sentences, and the *X axis* represents the error probability of the modified version of the sentences.

The first plot from left to right in Figures 3.5 and 3.6 shows that all the errors affect similarly to the Embedding featurization. Nevertheless, the error that affects the most is the *confused word error* followed by *extra word error* and *missing word error*.

The second plot in Figures 3.5 and 3.6 shows how POS featurization is similarly affected by *extra word error* and *missing word error*, and slightly less affected by *confused word error*. This result responds to the expectations since the distance between every pair of one-hotted POS tags is the same, and occasionally the confused word can have the same POS tag as the original word.

Finally, Pause featurization seems to be unaffected by *confused word error* and *extra word error*, but it is affected by *missing word error*. This is also expected since *missing word error* and *confused word error* do not alter the pause between the duration of words, while *extra word error* creates a pause between the two words added.

[8] https://pypi.org/project/fastdtw/

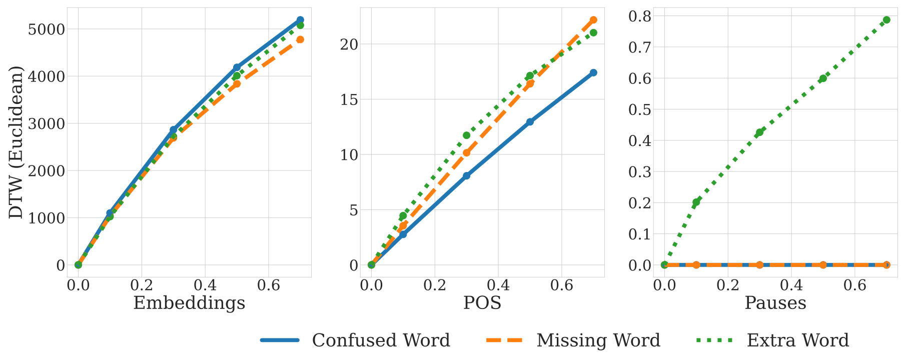


Fig. 3.5: DTW distances between 1000 Switchboard sentences and their modified versions generated by the ASR-SIM.

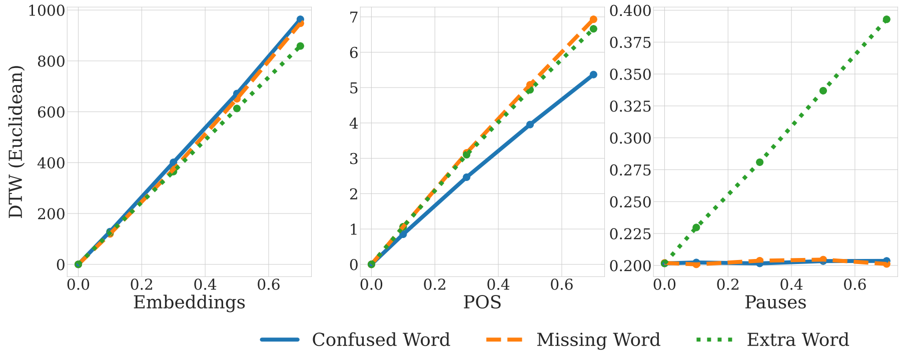


Fig. 3.6: DTW distances between 1000 subtitle sentences and their modified versions generated by the ASR-SIM.

3.5.2 Effects of the ASR-SIM errors on EOTD-M

We compute the predictions made by the LSTM based classifier given different errors in the ASR-SIM transcription, different sentence featurization techniques and different scenarios. The results are shown in Figures 3.7, 3.8, 3.9 and 3.10.

For each dataset, there are two figures displaying the results for the *same_distribution* and *different_distribution* strategies. Each of the Figures 3.7-3.10 is composed of three plots, one for each ASR-SIM error type defined in Section 3.2. Each plot shows the average AUC score obtained from a 10-fold experiment for each featurization technique and the combination of the three techniques, taking into account the variability obtained from randomly generating dataset splits and error generation.

A first analysis of Figures 3.7-3.10 reveals that the Pause feature representation obtains the worst AUC results not only for every error probability on the *same_distribution* experiments, but also for low error probabilities on the *different_distribution* experiments. This poor performance can be explained by the fact that pause duration information is very limited and does not capture semantic aspects of EOTD-M. This seems to be confirmed by the observation that the Combined features produce the highest AUC values, being the most complex representation used in this work.

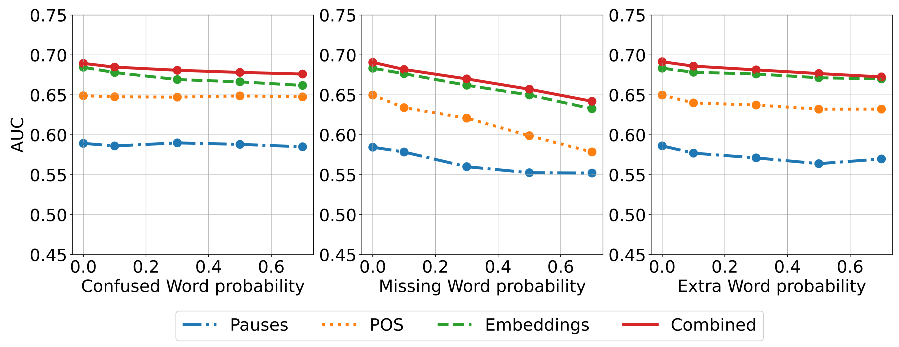


Fig. 3.7: Results for the Switchboard dataset with equal train-test distribution.

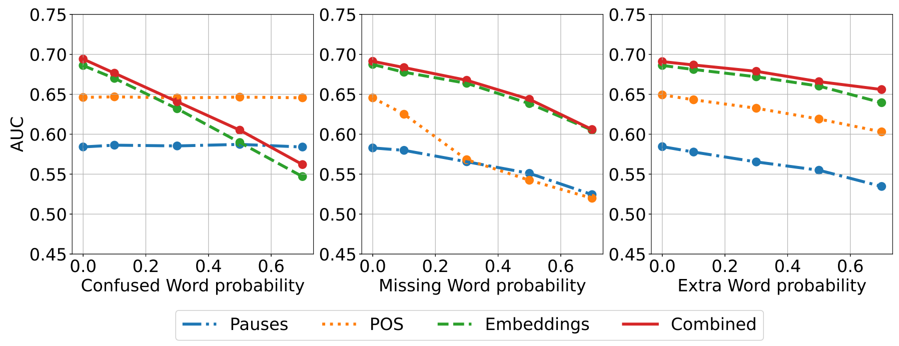


Fig. 3.8: Results for the Switchboard dataset with different train-test distribution.

Therefore, we focus our analysis on the Combined, Embeddings and POS features since, as previously discussed, Pauses features do not produce accurate classifications. Analyzing the effect of the shift of error distributions between train and test sets (*same_distribution* vs *different_distribution*), Figures 3.7-3.10 show that the effect of the change of distributions is remarkable under the effect of the *confused word error* and *missing word error*, and, to a lesser extent, for the *extra word error*. This effect is similar in both datasets, and more remarkable in the Combined and Embeddings experiments, which show fast degradation as the error probabilities grow. The payoff of having the most complex features is that it is the most sensitive to errors, deteriorating to the point of performing worse than other simpler features. This can be appreciated in *Confused* and *Missing word error* of Switchboard results in Figure 3.8, and on the Subtitles results shown in Figure 3.10.

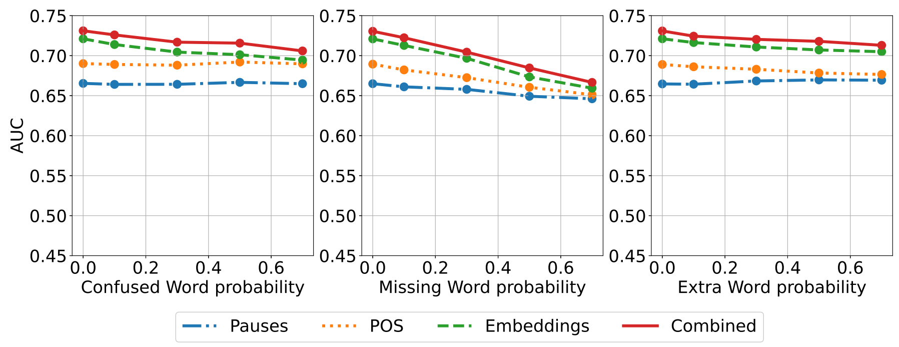


Fig. 3.9: Results for the Subtitles dataset with equal train-test distribution.

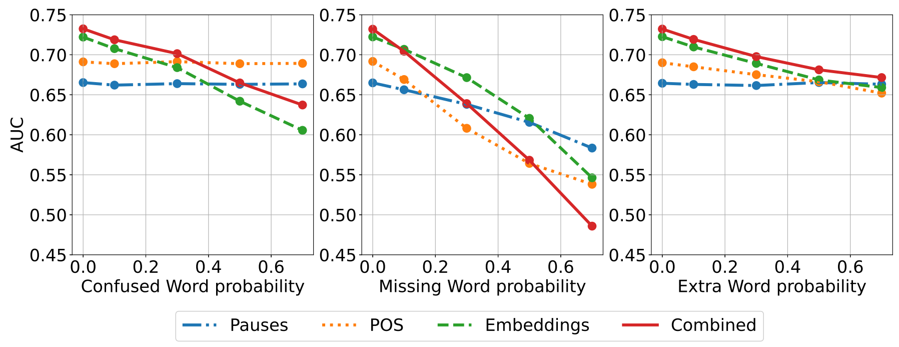


Fig. 3.10: Results for the Subtitles dataset with different train-test distribution.

On the other hand, POS features, despite achieving a worse performance than the Embedding features, are less affected by low error probabilities. Nevertheless, *missing word error*s have a stronger impact than the other errors, as can be seen in Figures 3.8 and 3.10. We may hypothesize that a confused word can still keep the same POS tag, and does not alter the vectorized representation of the sentence as much as a *missing word error*. In the same way, extra word errors generate two words, of which at least one could also have the same POS tag as the original word, despite having an extra tag from the other word. This hypothesis is reinforced by the results illustrated in Figures 3.5 and 3.6, and analyzed in Section 3.5.1, where *confused word error* is the least severe error in terms of altering the POS featurization of a sentence.

The impact of these induced errors has been also measured in terms of training time. In figures 3.11-3.12 the training time under each error probability is illustrated. In each figure, given the featurization technique, we can compare the training time for each error type. The X axis in this type of plots (Swarm plots and violin plots) does not correspond to a continuous variable, it acts as an auxiliary variable that helps to plot multiple instances that have the same Y value (training time in this case) without overlapping those instances. These figures show how, for both datasets, neither the error type nor the probability affect the training time significantly. Although the lack of influence of the errors on the training time could be caused by the 200 epoch limit by making all the training processes stop at the same epoch, this is not the case since none of the training processes reached the epoch limit.

Summarizing all the information extracted, we can conclude that *missing word error* is the most potentially harmful error an ASR-M can deliver to the EOTD-M if the classifier is not trained with the expected error probabilities. Not only it modifies the vectorization of a sentence severely, but it is also the error that affects the performance of an LSTM based EOTD-M the most. Another important finding is that representations that are less efficient for the EOTD-M under low error probabilities can become more efficient for particular types of errors when the error rate is increased. This is the case of the POS representation, which can outperform the embedding representation for high *confused word and extra word error* probabilities in the different train-test distribution scenario. Nevertheless, for this to happen in some embedding and error type configurations, the error probability must reach values of 0.5 or above, such is the case of Figure 3.8 for *confused word error*. In other studies such as [113] and [98], where the effects of *confused word error* on embeddings and NLU-M respectively are studied, the maximum error probability simulated is 0.5, and real transcription errors, the percentage error was 23%. Nevertheless, in this study we have covered a wider range of error probabilities, since the amount of errors depend not only on the ASR-M itself but also the audio conditions, and that is sometimes an uncontrollable factor.

The results obtained from the experiments are similar using both datasets, and the behavior of the classifiers under the influence of the generated errors

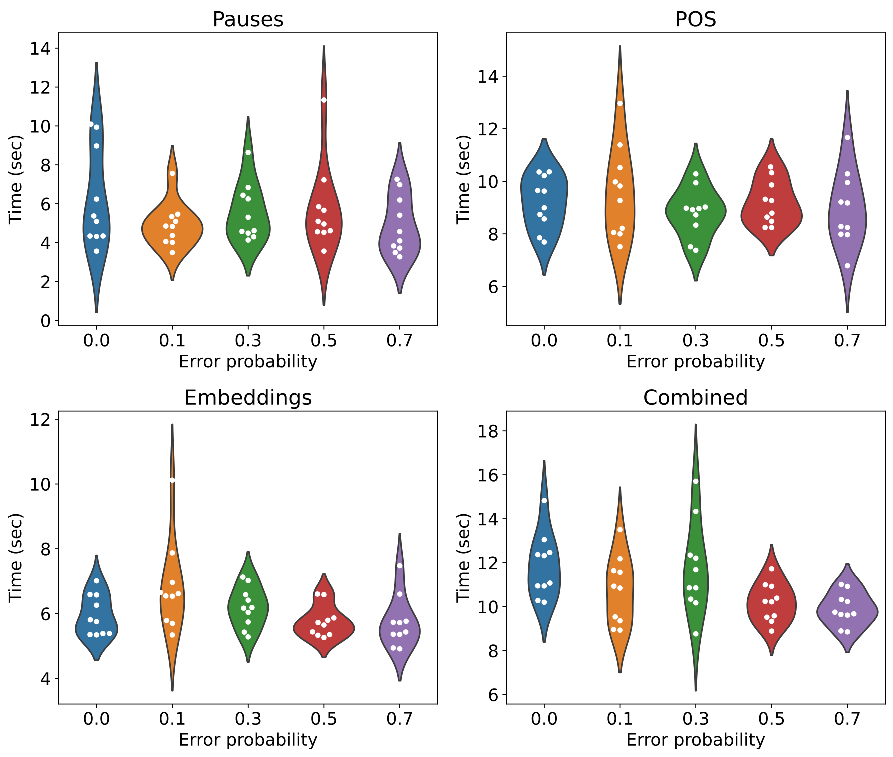


Fig. 3.11: Training time for the Switchboard dataset.

is coherent. This indicates that the ASR-SIM is suitable for the purpose of simulating ASR-M transcriptions and simulating errors.

3.6 Conclusions and future work

In this work we have proposed a method for investigating the influence of the ASR-M output errors on the behavior of the EOTD-M of Spoken Dialog Systems, which has not been addressed before. The ASR-SIM introduced in this work generates the transcription of a simulated dialog starting from plain text, with the amount and type of noise specified by the user. This leads to a realistic simulation of a variety of problems exhibited by ASR-M components. The code of this simulator will remain available in GitHub [9] for future studies as a contribution of this work. The absence of comparable simulators in the literature, is one of the motivations of this work.

Some of the insights from our analysis are the following:

[9] https://github.com/CesarMontenegro/AsrSimulator

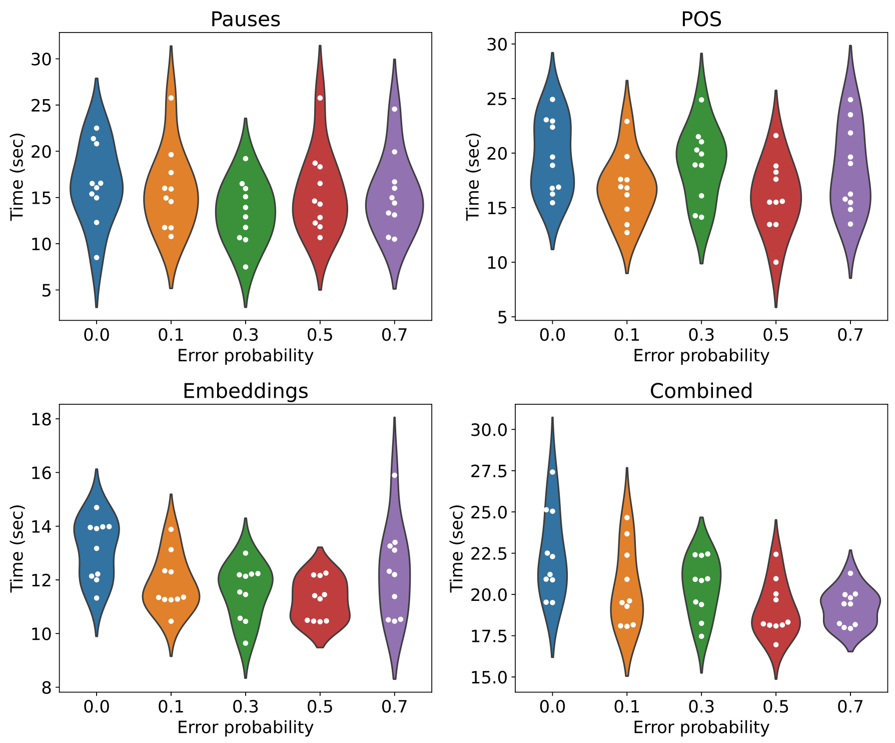


Fig. 3.12: Training time for the Subtitles dataset.

1. The ASR-SIM is suitable for the purpose of simulating ASR-M transcriptions and simulating errors.
2. Word embeddings produce the best overall results for the EOTD-M task. This is consistent with previous reported results.
3. The most influential error across representations is the *missing word error*.
4. In terms of classifier performance, there is an interaction between types of errors and featurization techniques.
5. It is more effective to include ASR-M simulated errors in the train and validation sets in order to make the classifiers more robust.

So far, we have only exploited the capability of the ASR-SIM to vary the three types of noise exposed. However, further research can be done by combining noise with the different speech profiles described in Section 3.2.1. Moreover, in this early version of the ASR-SIM, errors are generated randomly among the words of a sentence, nevertheless there are probably certain characteristics in some words that make them more prone to errors compare to others. A study on what word characteristics are more influential on the probability of a word to be miss transcribed would help to create more realistic scenarios by the ASR-SIM. Also, in order to increase the amount of algorithms that can benefit from this simulator, the pause and word duration simulations can be extended with other simulations, such as tone and other variables extracted from audio. Doing this, solutions based in the architecture represented in Figure 3.1b will be able to benefit from the advantages that the ASR-SIM offers.

4

The EMPATHIC dialogue-act taxonomy

4.1 Introduction

In this section, we focus on an important building block for the conception of a DM of an intelligent VC: the definition of a dialog-act taxonomy for implementing the communication between the intelligent agent and the user. One particularity of our virtual agent is that the dialog management implements a coaching model that is aimed at assisting elder people to keep a healthy and independent life. Therefore, the dialog-act taxonomy should take into account the coaching goals of the agent as well as the particular characteristics of this population segment. Furthermore, in contrast to other applications where the conversation is guided by the user's intents, here it is the agent the one that should guide the conversation to achieve a number of coaching objectives.

A dialog-act taxonomy for this type of systems should provide a robust platform to capture the semantics of the wide range of aspects involved in the agent-user interaction: such as providing assistance in daily tasks, suggesting health improving routines, and promoting social interactions. All these objectives require to carefully select the dialog acts and the way they are organized.

User utterances are classified according to a previously defined set of dialog act types, which may consist in a set of semantic units representing the translation of words into concepts. This is the work of the Natural Language Understanding (NLU) module shown in Figure 1.1. NLU is used to denote the task of understanding natural language coming from speech, conversation or other sources. In this work it is included in a spoken dialog system, so it denotes the task of understanding the natural language of a human in a conversational human-machine interaction. Therefore, the definition of the act tag set or dialog act taxonomy that will serve to label the dialog corpus, for both the output of the NLU module and the input of the dialog manager, is a critical step.

Several works have addressed the question of defining dialog act taxonomies [46, 109, 12, 101] that will be discussed in Section 4.3. Among them, the DIT++ taxonomy [10] and the more recent ISO 24617-2 standard [75, 13], which is intended to be a development of the previous one, can be considered the general methodological framework of the taxonomy defined in this work. In this framework, our aim is to develop a taxonomy conceived for a particular application: virtual coaching designed to keep a healthy and independent life as we age [52, 107]. This virtual coach is the main goal of the EMPATHIC project (Founded by the European Commission H2020-SC1-2017-RIA, grant number 769872). Studies suggest that attention to the lifestyle of the elderly can help them to maintain independent life [120]. In this application, the conversational agent is expected to guide the subject to pursue short and medium terms goals to promote healthy life style and social interaction, as well as to assist the user in the execution of daily tasks. As such, the conversational agent should be able to behave properly beyond task-specific domains.

We address the conception of such an agent from the perspective of coaching [27] which is a method that consists in accompanying, instructing, or training a person with the goal of achieving goals, or developing specific abilities. In our framework, coaching is focused on a reduced number of domains, i.e., nutrition, physical activity and social engagement. These are domains whose role is critical for keeping a healthy and independent life of elderly. As a consequence, the NLU system has to understand the user in terms of the coaching goals, thus, its output needs to be extremely related to these goals. Moreover, the dialog-act taxonomy has to allow the dialog manager to implement a strategy according also to the specific goals of the coaching model, in contrast with classical systems that need the decoding of the user intents. Additionally, the conversational agent is also expected to participate of more general conversation, i.e., chit-chat talk.

Thus, in brief, the main contributions of the work include the definition of a dialog-act taxonomy aimed to represent the user utterances in the particular human-machine communication framework of the EMPATHIC project, which develops a coaching model aimed at keeping a healthy and independent life of elderly. Thus, the taxonomy allows the Dialog Manager to understand the user in terms of the coaching strategies and goals to be developed and agreed with the user, which is a challenging and novel approach. In addition, a set of real human-machine interactions in Spanish between elderly and a simulated virtual coach, i.e., through Wizard of Oz (WoZ) experiments so that the Wizard plays the role of a coach, has been annotated and discussed, providing a preliminary validation of the proposed taxonomy as well as important cues for future work in the field. All in all resulting in an original contribution in terms of language (Spanish), framework (coaching) and target population (Elderly).

4.2 Wizard of Oz Method for data gathering

Various modules of the EMPATHIC VC will require ongoing development and improvement, e.g. to advance speech recognition and language understanding, or foster user experience and acceptance. In order to simulate such modules while they are being developed, the goal was to build and consequently use a simulated Wizard of Oz (WOZ) component. WOZ constitutes a prototyping method that uses a human operator (i.e., the so-called wizard) to simulate nonor only partly-existing system functions. In language-based interaction scenarios, like the ones envisioned by EMPATHIC, WOZ is usually used to explore user responses and the consequent handling of the dialog, to test different dialog strategies or simply to collect language resources (i.e., corpora) needed to train technology components. In EMPATHIC, however, the goal is to use WOZ beyond this traditional prototyping stage, and make it a fallback safety net for situations in which the automated coach may be unable to respond. That is, the goal is to develop a system component, which initially serves as a prototyping tool supporting the research on language-based interaction and dialog policies, but then becomes an always-on backup channel dealing with those user requests the system is incapable of handling by itself. A first version of this tool has been built and consequently used in several user studies

4.3 Related Work

While the GROW model serves as a conceptual pillar for developing the dialog-act taxonomy, we also look to previous approaches for dialog-act tagging.

Coding a sentence with a set of labels goes back to speech act theory of Austin [7], which has been the basis for modern data-driven dialog act theory. Multiple different dialog act taxonomies have been proposed to solve the task of assigning dialog act labels to sentences. They not only differ in the precise set of tags selected, but also in characteristics such as whether the tags are exclusive, level of detail or structure.

Dialog act taxonomies can be characterized taking into consideration different criteria, such as the following:

- Type of communication (i.e., synchronous vs. asynchronous).
- Activity type and dialog domain.
- Type of corpora (e.g., speech dialogs, videos, chat).
- Types of speech act classification schemes.
- Dimensions (unidimensional versus multidimensional annotation).
- Annotation tools and annotation procedure.

Books, and other written forms of communication, are asynchronous methods of communication where each message is thought beforehand. This generally gives written communication a better structure than spoken communication, where doubts, rectifications, and external factors such as noise or user speech characteristics, may result into incomplete or fuzzy messages. In this context, the PDTB [78] taxonomy was designed for annotation of discourse relations between sentences, analyzing the conjunctions used to relate them. The sense tags described in PDTB have a hierarchical structure, but they would not suit to our coaching problem since they are designed to deal with asynchronous communication.

In a human to human conversation, these problems are solved by considering the context of the conversation. Dialog acts need to take into account whether the communication is synchronous or asynchronous. For instance, synchronous communications allow the introduction of clarification intents, where an agent may be instructed to repeat a question or formulate it in a different manner. Such type of intent tags make no sense for asynchronous methods. The dialog-act taxonomy introduced in this work has been conceived for synchronous communication.

While one of the most common applications of act labeling is in the context of human to human or agent to human conversation, there are other types of activities to which they have been applied [76]; for instance, they can be used for text summarization [128]. Similarly, there is a variety of platforms and domains of applications to which act labeling methods have been applied, such as social networks [128], and classification of message board posts [80]. The proposal we introduce in this work is oriented to represent spoken communication between an agent and a human.

Another important difference between dialog act-taxonomies are the corpora to which they are applied and from which machine learning models are commonly learned. The corpus used is, most of the time, strongly related to the domain in which the dialog act are going to be applied, and therefore should be able to capture the particularities of the domain. Initially, available corpora were mainly created from task-oriented dialogs [5]. More recently, larger corpora have been proposed for training end-to-end dialog systems [53, 129]. In general, these large corpora are not annotated.For a survey on available corpora for dialog systems [93] can be consulted. The corpus used in this work has as a particular characteristic, the fact of being obtained from elderly people, a social group for which dialogs are more scarce.

Among the dialog-act models proposed in the literature, the approach introduced in [101] presents a framework to model dialogs in conversational speech. The dialog act taxonomy was first based on a set of tags that was used for the annotation of the discourse structure and then modified to make it more relevant for the target corpus (the Switchboard corpus [26]) and the task. However, existing dialog act taxonomies were not designed for the scenario described in Section 4.4.1, where a virtual agent is the responsible for the development of the conversation. DAMSL taxonomy [4] was developed primarily for two-agent task-oriented dialogs. Nevertheless, the Empathic taxonomy has some common features with DAMSL, since the Intent dimension defined for Empathic taxonomy contains labels related to 3 out of 4 DAMSL categories (Information level, Forward Looking Function, Backward Looking Function).

Our proposal for the Empathic project has aspects in common with DIT++ [11], although they are designed for different types of interactions. On the one hand, DIT++ is based on traditional task-oriented conversations, on the other hand Empathic is based on coaching interactions, where the agent is an active member of the conversation from the point of view that the coach guides the conversation throughout the GROW model strategy. Nevertheless, many of the labels present in the taxonomy of general-purpose functions and dimension-specific functions defined in [11] can also be found in the intent label defined for Empathic. Another work relevant for our approach is the one recently published in [68], where a hierarchical schema for dialog representation is proposed. Although the introduced scheme is specifically conceived to support computational applications, it uses a structure of linked units of intent that resembles the hierarchical structure at the core of our proposal.

The common norm for dialog act annotation is that a single communicative function is assigned to an utterance. However, some works propose multidimensional dialog act taxonomies in which multiple communicative functions may be assigned to the utterances. DAMSL considers a set of exclusive group tags as different dimensions, whereas DIT++ considers a dimension in a multidimensional system, as independent, and can be addressed independently from other dimensions. In particular, a 9-dimensional annotation scheme was defined in [12]. Similarly, we use a multidimensional taxonomy which allows us to capture richer semantic information from the dialogs. Considering multiple modes of semantic information is a requirement for implementing an agent that should be able to embed the coaching objectives as part of the dialog strategies. Without the rich information provided by multiple types of tags, it would be very difficult to guide the user to the satisfaction of the objectives, and to evaluate whether these objectives have been fulfilled. The details of this multi-modal taxonomy are described in Section 4.5.

Although the multidimension criteria is similar in both taxonomies, DIT++ is designed for turn labeling, and Empathic is focused in subsentence labeling. This difference forces DIT++ to separate into two dimensions different intents that can be found in the same turn as seen in the dialog examples found in [41]. Labeling subsentences allows the Empathic taxonomy to group aspects found in the DIT++ dimensions defined in [11] (general-purpose functions, dimension-specific functions), since a turn will be split into subsentences, having only one intent label for each one, avoiding the problem of having two intent labels in the same turn.

In addition, an important effort has been carried out to define the ISO 24617-2 standard [75, 13, 14] that includes the 9 dimensions defined in DIT++ and reduces the number of communicative functions, which can be specific for a particular dimension or general-purpose communicative functions that can be applied in any dimension. In addition, the standard also considers different qualifiers for the certainty or the sentiment. This approach has also be included in our proposal, which can be considered, to some extent, as a reduced and GROW-driven adaptation of main characteristics of DIT++ and

ISO 24617-2 for the Empathic purposes. An additional aim of the standard is to produce interoperable annotated dialog resources. To this end, a set of dialogs from variety of corpora and dialog annotation schemes, such as the Map task, Switchboard, TRAINs or DBOX, have been re-annotated under ISO 24617-2 scheme to build a Dialog Bank [13].

Finally, proposals for dialog act taxonomies also differ in the annotation tools used and annotation procedures. Humans are better than machines at understanding and annotating dialog utterances in a detailed manner, because they have more knowledge of intentional behaviour and they have richer context models [12]. So we rely on human annotation procedures to get accurate annotations instead of using automatic methods. We explain the characteristics of our annotation procedure in Section 4.6.

Regarding the NLU task, having a semantic representation that is both broad coverage and simple enough to be applicable to several different tasks and domains is challenging. Thus, most NLU system approaches depend on the application and the environment they have been designed for. In this way, targeted NLU systems are based on frames that capture the semantics of a user utterance or query. The semantic parsing of input utterances in NLU typically consists of three tasks: domain classification (what is the user talking about, e.g., "travel"), intent determination (what does the user want to do, e.g., book a hoter room) and slot filling (what are the parameters of this task e.g., "two bedroom suite near disneyland") [111]. The domain detection and intent determination tasks have been typically treated as semantic utterance classification problems [124, 35]. Slot filling, instead, has been treated as a sequence classification problem in which semantic class labels are assigned to contiguous sequences of words [116], which is now addressed by bidirectional LSTM/GRU models among others [33, 114]. A good review of the NLU evolution is given in [110].

While the NLU employed in this work does perform intent detection and adds entity recognition, as other approaches do, the taxonomy includes intent labels specifically conceived for the GROW model, which has to fulfil additional objectives. For example, the taxonomy has to provide a relationship among the user utterances and the goals of the GROW model, which has to be agreed between user and virtual agent, and therefore be established, during the conversation, as mentioned in Section 4.4.1.

4.4 Dialog Acts for an Empathetic Agent

The conversational agent, as part of the project described in [52], faces several novel challenges. It will work on real time, the utterances will be automatically extracted from speech using an Automatic Speech Recognition (ASR) module, and the agent is expected to understand and produce three languages (namely Spanish, Norwegian and French, abut also English, German and Italian for research support). But notice that the results presented in this work are for experiments conducted in Spanish. In addition, the agent will perform a variety of tasks to analyze the development of the conversations with the user.

Figure 1.1 illustrates just some of the main modules in the software, and the flow of information as designed. The modules *Natural Language Understanding* and *Dialog Manager*, highlighted in blue, are the modules affected by the dialog act taxonomy definition presented in this work. The dialog act set has been conceived taking into consideration these challenges and its design and description is the focus of this paper.

4.4.1 GROW Model Implemented through the Dialog Manager

The Dialog Management (DM) is a fundamental component of any Spoken Dialog System (SDS). It maintains the state and manages the flow of the conversation by determining the action that the system has to perform at each agent turn. For the EMPATHIC project [107, 106], we used an agenda-based management structure based on the RavenClaw [9] dialog management framework that separates the domain-dependent and the domain-independent components of the conversation, unlike previous plan-based dialog managers. The domain-specific aspects are defined by a dialog task specification defined by a tree of dialog agents. Then a domain-independent dialog engine executes any specified task using a stack structure to control the dialog while providing reusable conversational skills, such as error recovering. This approach is suitable for dealing with complex domains while allowing the use of a relatively unconstrained natural language.

The DM and the involved strategy implement the coaching model chosen for the project. Coaching has been defined as a result-orientated, systematic process. Coaching generally uses strong questions in order that people discover their own abilities and draw on their own resources. In other words, the role of a coach is to foster change by facilitating a coaches' movement through a self-regulatory cycle [28]. There is evidence showing that coaching interventions can be effectively applied as a change methodology [104, 39]. One of the most common used coaching methodologies is the GROW Model [119]. This model provides a simple methodology and an adaptable structure for coaching sessions. Moreover, efficiency has been demonstrated in some Theoretical Behavior Change Models such as the Trans theoretical Model of Change (TTM) [71, 70]. As a consequence, this coaching model has been selected for the EMPATHIC project to be integrated in the DM strategy.

A GROW coaching dialog consists of four phases which give the name to the model: Goals or objectives, Reality, Options and Will or action plan. During the first phase, the dialog aims at getting the specification of the objective that the user wants to achieve, for example, to reduce the amount of salt in order to diminish the related risk of hypertension. Then, this goal has to be placed within the personal context in which the user lives, and the potential obstacles need to be identified. In the next phase, the agent goal is to make the user analyse the options he/she has to achieve the objective within his/her reality. Then the final goal of the dialog is the specification of an action plan that the user will carry out in order to advance towards goals. In this framework, the DM strategy also involves achieving the goals associated with each of the four stages. First, it will try to get a specific goal from the user, asking something like ("Would you like to improve something in your eating habits?"). Once the user provides a sentence including his/her goal, the DM will focus on the next stage. Thus, it will try to get information about the context in which the goal has to be achieved, asking something like ("How often do you usually go to the grocery?"). In this way the dialog will be developed until all the stages are completed. This strategy, differs from classical task-oriented dialog systems in which user asks something related to the task, and then the system tries to obtain additional information, if needed, to be able to provide as accurate a response as possible. In fact, the particular user goal and related action plan have to be agreed between the virtual agent and the user during the conversation. However, this strategy can still be correctly specified by the Ravenclaw domain dependent trees of dialog agents mentioned above that define the domain specific aspects of the dialog.

The EMPATHIC virtual coach is planned to deal with four coaching subdomains: nutrition [91], physical activity [92], leisure [90] and social and family engagement.

4.5 Proposed Dialog Act Taxonomy

As discussed in previous section, the characteristics of the taxonomy must be defined according to the conversational agent needed. In our case, the agent must maintain conversations in real time about a reduced set of topics, and follow coaching strategies to guide the user. Instead of displaying a passive or merely reactive attitude, it should be pro-active and assertive, proposing different activities and topics of conversation to the elder user. In order to mitigate the difficulties in automatic labeling, due to the reasons explained, we propose a multidimensional hierarchical taxonomy to represent the relationships between the tags. Four types of labels are used, *Topic*, *Intent*, *Polarity*, and *Entity* labels.

The *Topic* label classifies the utterance in a number of classes relevant to determine the general context in which the conversation is framed. Tracking the *Topic* label will help the conversational agent to detect when the user is changing the subject of the conversation. Due to the links between the target topics the conversational agent is designed for, it is common to switch from one topic to another. Nevertheless, the DM implements a GROW-based strategy, so in this work the DM has its own goals according to the GROW model. In this framework the *Topic* label will assist the DM to associate the user utterance to both, the user and the DM goals, which have to be agreed during the interaction.

The *Intent* label classifies the utterance in classes related to the user's communicative intentions (e.g., *question*, *inform*, etc.). Our particular choice of the *Intent* labels is based on the GROW model of coaching, which is probably the best known session structure model [29]. The set of *Intent* labels we have defined is aimed to help the conversational agent to detect Goals, Realities, Obstacles and Ways forward of the particular topics the agent has been designed to deal with.

The *Polarity* label aims at representing the sentiment associated to the semantics of the user turn, which can be very relevant to provide exploitable information to dialog managers. We distinguish between three levels of polarity: positive, neutral and negative. This label is a product of the analysis of the text, as topic and intent labels, in contrast with emotions detected from the spoken language represented in Figure 1.1.

The *Entity* label is different to those ones previously described in the sense that it does not serve to classify an utterance. Instead, it is applied to classify particular elements that provide specific semantic information about the Intent label. However, since the particular set of entity labels that we have selected play an important role in the semantic analysis, we consider it as a fourth modality, together with the other three. Also, entities can be useful to improve the naturalness of the conversation, when the names of the relatives are detected, or used to formulate specific proposals.

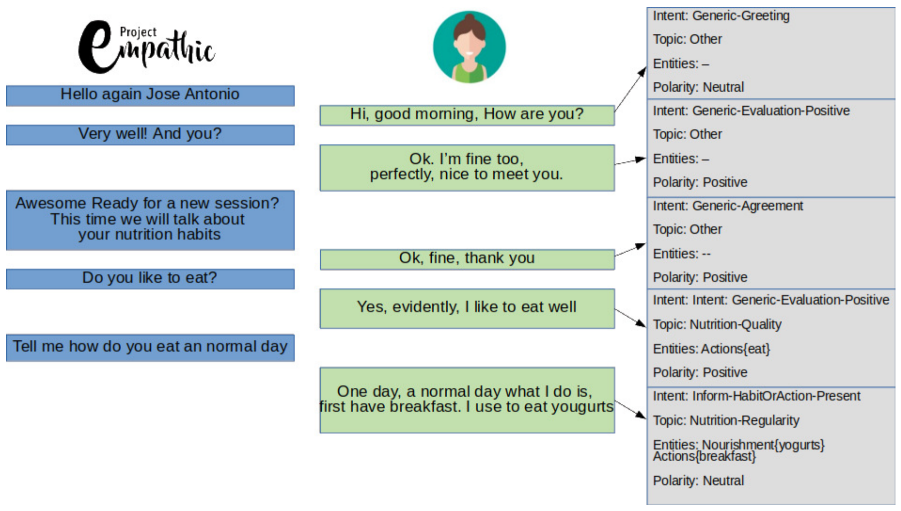
For *Topic* and *Intent* labels we propose a hierarchical structure. This means that an utterance is labeled by multiple tags that can be ordered from more general to more specific. Such labeling can be graphically represented using a tree. In this structure, the closer a label is to a leaf, the more precise it is, while the further away from the leaves, the more general. Figure .1 in Appendix shows the topic label tag set organized as a tree. Four main groups can be recognized: *nutrition*, *sport and leisure*, *family engagement* and *other*. Each of these groups further splits into more detailed categories. Similarly, Figure .2 shows the hierarchical structure for the Intent tags. Finally, in Table 4.1, the entities are shown.

The rationale behind the use of hierarchical labels is to allow the agent to receive more fine-grain information when possible, but still useful less refined classification when no other choice is available. Hierarchical structures allow the experts to add more knowledge during the labeling of a dialog corpus. In addition, when the automatic labeling model is trained, it can be less precise at the time of making predictions in those situations in which the confidence is not high enough to discriminate between two labels, selecting the parent label. This ambiguity, permits the conversational agent to guess depending on what it is expected and taking into account the other labels available. In addition, it allows the virtual coach to understand the user in terms of the system goals and topics, and thus to keep the control of the dialog. Also, it permits the conversational agent to formulate specific questions to solve the ambiguity.

Table 4.1: List of Entity categories.

Persons	Relatives	Objects/Utensils
Actions	Nourishment	Sport and leisure
Books	Cardinal numbers	Music/Bands
Quantities	Ordinal numbers	Films/TV Series
Frequencies	Time amount	Paintings/Sculpture/Art
Diseases	Absolute dates	Places, buildings and organizations
Emotions	Relative dates	Nationalities
Meteorology		

In Figure 4.1, a labeling example is illustrated for the Spanish corpus. Its translation to English is shown in Figure 4.1. This example has been labeled by a human, and even though we have context information, sometimes it is impossible to reach the tree leaves. In the ambiguous example, it is not possible to set a more specific topic label than *nutrition*, although with context information, we could deduce that the user is referring to the little amount or variety of fruit he or she eats.



Hello again Jose Antonio

Hi, good morning, How are you?

Very well! And you?

Ok. I'm fine too, perfectly, nice to meet you.

Awesome Ready for a new session? This time we will talk about your nutrition habits

Ok, fine, thank you

Do you like to eat?

Yes, evidently, I like to eat well

Tell me how do you eat an normal day

One day, a normal day what I do is, first have breakfast. I use to eat yougurts

Intent: Generic-Greeting
Topic: Other
Entities: –
Polarity: Neutral

Intent: Generic-Evaluation-Positive
Topic: Other
Entities: –
Polarity: Positive

Intent: Generic-Agreement
Topic: Other
Entities: --
Polarity: Positive

Intent: Intent: Generic-Evaluation-Positive
Topic: Nutrition-Quality
Entities: Actions{eat}
Polarity: Positive

Intent: Inform-HabitOrAction-Present
Topic: Nutrition-Regularity
Entities: Nourishment{yogurts} Actions{breakfast}
Polarity: Neutral

Fig. 4.1: English conversation example.

4.6 Using the Taxonomy to Get a Labelled Corpus

4.6.1 Annotation Procedure

The proposed taxonomy was applied to label the user turn of the human-machine set of conversations acquired through the Wizard of Oz technique in the EMPATHIC project. To this end two scenarios were chosen. The first is an introductory and quite open dialog, where the machine presents itself and asks the user about his or her hobbies. The main goal of this scenario was to make the participant feel comfortable while interacting with the simulated virtual agent. The dialogs of the second type implement a coaching session in the area of nutrition, according to the GROW model. Dialogs were acquired in three different countries with different language and culture: Spain, France and Norway. In total, 192 elder participants are interacting with the system, 72 in Spain and 60 in both France and Norway. Every user speaks with the machine in the two scenarios, and thus the final corpus will consist in 384 dialogs. Each dialog is approximately 10 minutes long, which results in an average of 30 turns per dialog.

For the moment only the Spanish dialogs have been annotated according to the procedure shown in this work. To do so, 9 different annotators were instructed about the labels, the GROW model, and about the context of the project. Each of the annotators labeled roughly the same number of dialogs. Each dialog was labeled by only one annotator. Nevertheless, all the annotators worked together to deal with doubts and disagreements, under a close supervision of the first and the second author of the paper, resulting in a collaborative annotation task. Each annotator labeled dialogs corresponding to both the introduction and the nutrition scenarios. Table 4.2 shows the main numbers of the annotated corpus.

Table 4.2: Description of the annotated corpus.

Characteristics	**Number**
Number of users	72
Number of dialogs	142
Number of turns	4522
Number of running words	$72,350$
Vocabulary size	5543
Number of topic labels	55
Number of intent labels	34
Number of running entities	11,113

Since more than one intent and topic can appear per turn, we asked the annotators to divide each turn into subsentences that roughly correspond to uttered clauses, so unique intent and topic labels can be assigned to each of these subsentences. To do so and to carry out the annotation procedure, we developed an annotation tool that provides a simple command-line interface. This tool shows all the user turns in a dialog sequentially. For each of them it first asks to identify the entities. Then the annotator splits the user turn into the subsentences. Finally he or she selects, for each subsentence, the topic, intent and polarity labels. The annotators took around an hour to label each dialog, on average.

After the annotation process, each turn was divided into 1.92 subsentences on average. The left-hand side of Figure 4.2 shows a histogram of the number of subsentences that resulted from the splitting of turns. On the other hand, it also shows the distribution of the number of tokens (words and punctuation marks) per subsentence. These figures show a low number of sentences per turn as well as a low number of tokens per clause or sub-sentence. These distributions are consistent to human-machine interactions where there is a significant number of user turns just consisting on two or three words that express agreement, i.e., **yes**, or disagreement, i.e., **no**.

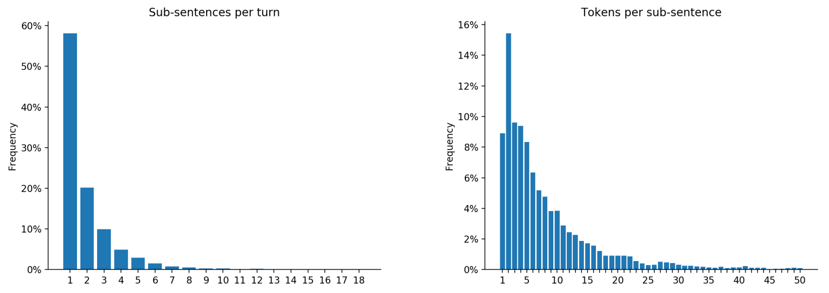


Fig. 4.2: (**Left**) Number of subsentences per turn. (**Right**) Number of tokens per subsentence.

4.7 Analysis and Discussion

In this section we will show the results and statistics related to these annotations. We will first focus on general statistics of the acquired dialogs and we will then dig into the annotation results. Table 4.3 shows the frequency of the most frequent topics appearing in the data. The sets of frequent labels are much more reduced than the sets of all possible labels shown in Figure .1 in the Appendix. The main reason for this is that even though the trees in Figures .1 and .2 were designed for the whole task of the EMPATHIC Project, the acquired data corresponds only to the explained two scenarios: the introduction and the nutrition scenario. As a consequence, a significant number of labels are under the *nutrition* domain whereas *hobbies* and *travelling* are associated to the welcome or introductory scenario. The label *other* includes the less frequent sub-labels as well as clauses that cannot be classified in terms of topics, such as generic agreement or disagreements. In the same way, Table 4.4 shows the frequency of the most frequent *Intent* labels. This table shows a significant incidence of the GROW related sub-trees. Thus the taxonomy proposed to represent the GROW model has demonstrated to be able to cover real users interactions with a Wizard who plays the role of a

Virtual Coach. In the same way, the high number of generic communication tools expressing opinion, as well as agreement or disagreement, depict well spontaneous human-machine conversations. This table also shows a certain positive attitude of the participants versus the virtual agent.

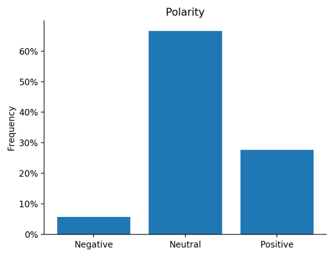
Table 4.3: Frequencies of the most frequent topic labels. The sets marked with the symbol * include all the labels under a given label, and also the cases where the annotator has not selected any sub-label.

Frequent *Topic* Labels	**Frequency**
nutrition	16.5%
sport and leisure- hobbies	5.9%
sport and leisure - travelling	5.8%
*sport and leisure - **	8.1%
Other	63.7%

Table 4.4: Frequencies of the most frequent intent labels. The sets marked with the symbol * include all the labels under a given label, and also the cases where the annotator has not selected any sub-label.

Frequent *Intent* Labels	**Frequency**
generic - agreement	17.4%
generic - disagreement	4.8%
generic - evaluation/opinion	18.1%
generic - doubt	3.3%
generic - greeting	4.0%
*generic - **	6.2%
GROW inform - habit or action	16.7%
GROW inform - objective	2.5%
GROW inform - obstacle	2.9%
*GROW inform - **	6.8%
question	3.5%
other	13.8%

Then, the left-hand side of Figure 4.3 contains the distribution of the polarity labels. As might be expected, the user is often neutral, sometimes positive and only rarely negative. This table is quite consistent with the outcomes of Table 4.4. Further analysis will need to be carried out to determine the correlations between these labels with the valence labels obtained through the emotion annotations based on speech and also on facial expressions.



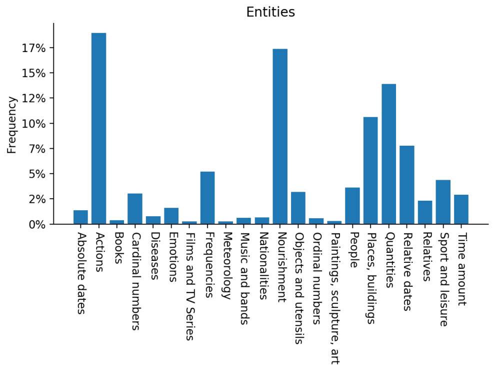


Fig. 4.3: (**Left**) Distribution of the polarity in each subsentence. (**Right**) Distribution of the entities.

The entities were identified at the turn level. An average of 2.6 entities were labeled per user turn. The right-hand side of Figure 4.3 shows the frequency of each of the entities. This figure shows a significant occurrence of entities that correspond to user utterances developing the coaching model proposed by the Wizard, such as *Nourishment* and *Actions*.

In Figure 4.4, we illustrate the relationship between topics and intents, by means of Sankey diagram (`https://en.wikipedia.org/wiki/Sankey_diagram`). In this and the following figures, the most representative labels (in terms of appearance in the labeled conversations) of two label groups face each other. The flows that connect the labels from one side with the other, represent the amount of sentences that are labeled with the two connected labels. The labels that are not representative enough, are included in the parent label appended with a star. Also, only labels down to the second level of depth are contemplated, any deeper label is included in its second depth level parent node. In order to help understanding visually the tree structure, all the labels pending from a first level node will have the same color.

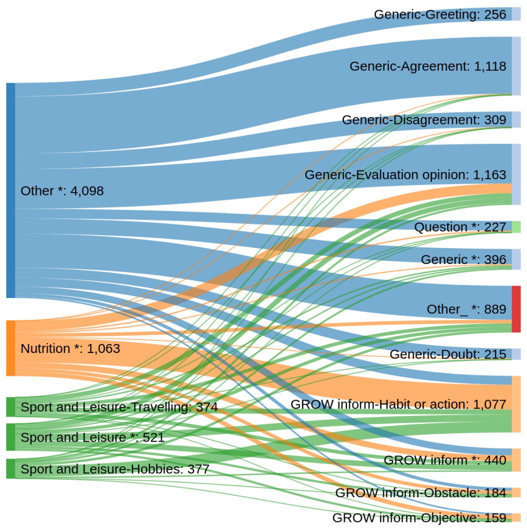


Fig. 4.4: Relationship between intent and topic labels.

Figure 4.4 also allows us a better understanding of the relation between both labelling as well. The most frequent *Topic* label is *other* with a 63.86% of the sentences. But it shows that this high percentage corresponds to general conversation, greetings, or answers to questions from the Virtual coach as seen in the flows that connect the *other* topic label with all the *generic* family of labels represented in blue. As the second session was planned to be about *Nutrition*, the Virtual coach had to ask about the user's habits. Therefore, a 16.69% of the intent labels are *GROW inform-habit or action*, what it was not expected was to have as much as *nutrition* habits explained as *sport and leisure-family habits*. Other intent labels share this particularity of having as much relation with *nutrition* as with *sport and leisure* as *generic-evaluation/opinion* and the *GROW inform* label family in light orange.

In Figures 4.5 and 4.6, Sankey diagrams are used to represent the relation between the entities and the intent and topic labels, respectively. For the sake of clarity these figures only represent the first level of *Intends* and *Topics* trees. Figure 4.5 shows how sentences that have *Nourishment* entities are often *GROW inform* sentences, what is consistent with the conclusions obtained from Figure 4.4 about the relationship between the *nutrition Topic* and the *GROW inform* labels. Also other entities are useful to inform about *nutrition* habits like *Actions*, *Quantities* and *Relative dates* for example.

The relations deduced in Figures 4.4 and 4.5, are reinforced with Figure 4.6, where we can also see the relation of *Nutrition* topic with *Nourishment*, *Quantities* and *Actions* entities among others.

The analysis presented and discussed in this section shows how a taxonomy developed from a theoretical coaching model, such as the GROW one proposed in this work, can be followed by real users interacting with the simulated agent. The annotated data show that the taxonomy is capable of fully coverage of the spontaneous utterances of the participants in terms of concepts, topics, communicative intends that are useful to provide meaningful information to the DM according to the the goals to be developed by the system, to a certain level of granularity, but still useful. In the same way, the distribution of the selected entities seems also to agree the goals of the developed scenarios. Thus, the outcomes of the data annotation could asses the hierarchy proposed, to some extent. On the other hand, the positive agreements and opinions, as well as the polarity distributions, seem to show a relaxed and positive attitude towards the Wizard, who played well the role of a coach. All in all these data seem to support the full procedure of the WoZ recordings.

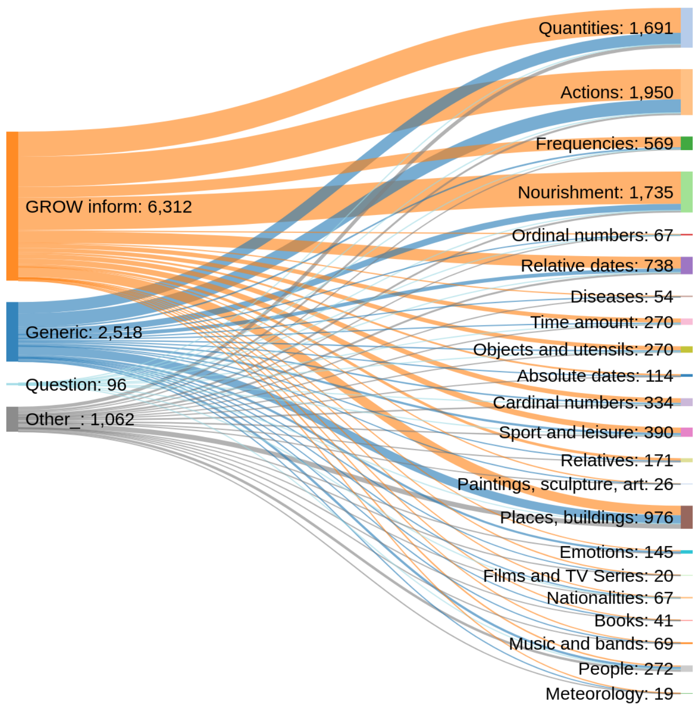


Fig. 4.5: Relation between intent and entities.

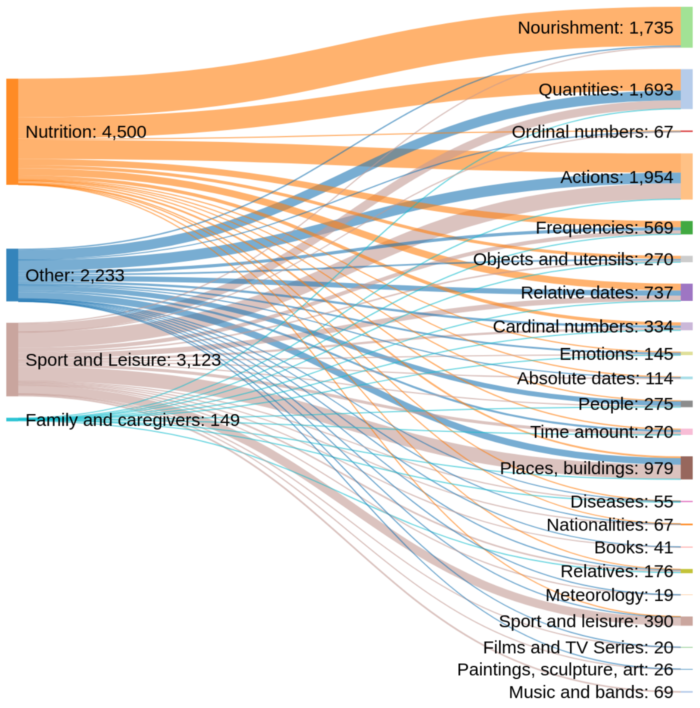


Fig. 4.6: Relation between topic and entities.

4.8 Conclusions

In this work we have introduced a dialog act taxonomy that has among its distinctive characteristics the capacity for supporting communication based on a coaching strategy, a hierarchical structure between the tags, and the fact of being multi-modal.

The coaching strategy is essential within the framework of EMPATHIC since it directly addresses the need to implement a pro-active agent that provides assistance and counseling to the elderly users, and drives the dialog with the intention of reaching coaching goals. This is an important difference to other approaches such as task-oriented dialog systems and chit-chat implementations.

The hierarchical structure allows us to capture varying degrees of semantic information from the utterances. Having different taxonomies for topics and intents allow the system a very rich semantic representation of the dialogs that provides more flexibility for the design of dialog managing strategies. Combined, these characteristics make our proposal significantly different to previous dialog act taxonomies and a very relevant proposal for the implementation of virtual agents.

Another important contribution from our work is to provide one of the first analysis of an annotated corpus constructed from 142 interactions between elder people and visual agents. This corpus is precious because it covers a population usually neglected in similar studies, mainly due to the difficulties involved in accessing to elderly and face them with the required technologies. We emphasize that the usefulness of the corpus, and of the obtained annotations, goes beyond the implementation of the virtual coach. Although the results shown in this work are limited to Dialogs in Spanish, there is ongoing work in the completion of annotated dialogs for the other two languages. Thus, further language and cultural comparisons will be achieved.

The validation of the introduced taxonomy will require the application of a classification strategy able to label the dialogs using the introduced sets of tags. The taxonomy could be indirectly evaluated in terms of the performance of the dialog manager that uses it. On the other hand, while hierarchical multimodal taxonomies, as the one we have introduced, are richer and provide much flexibility for the implementation of dialog managing strategies, they are also challenging for typical machine learning methods. For instance, hierarchical multi-label classification is more difficult that traditional multi-class classification problems. Similarly, using topic label information for implementing specialized coaching scenarios and switching between them is not a trivial task. Nevertheless, we consider that the taxonomy introduced in this work provides us with a good set of tools to face these challenges.

We foresee a number of ways in which the annotated data can be valuable for machine learning applications. While its size is relatively small (4522 turns for the Spanish corpus), this data is sufficient to refine machine learning models that had been trained using larger, more general, dialog corpora. In addition, as part of the EMPATHIC there will be annotated data for other languages. This fact points to the feasibility of obtaining a larger corpus by translating all dialogs to a base language. Moreover, it opens the possibility of investigating transfer learning strategies in a multi-lingual framework. We have already obtained preliminary results on the application of parallel corpora for training dialog classifiers (Montenegro et al. [63]). Finally, the annotated corpora is valuable itself due to the particular characteristics of the task as well as of the erderly population from which it has been obtained.

Finally, the analysis of the annotation data discussed in this work let to conclude that the taxonomy developed from a theoretical coaching model, such as the GROW model, has been able to provide fully coverage of the spontaneous utterances of real users interacting with a simulated agent who plays the role of a Virtual Coach. Moreover, this analysis shows significant frequencies of GROW related *Topic*, *Intends* and *Entities* labels. All in all, these outcomes could also provide a preliminary validation of the taxonomy proposed.

Part II

Contributions on Hierarchical Classification

5

Background

The taxonomy presented in Chapter 4 proposes two hierarchies for the labels of the Intent and Topic classification tasks. This taxonomy has led to the research presented in the following chapters, and extend beyond the scope of the EMPATHIC project and can find applications in other fields.

5.1 Hierarchical classification

HC can be seen as a particular type of structured classification, where the output of the classification algorithm is defined over a label taxonomy. The term structured classification is broader and denotes classification problems where there is some structure (hierarchical or not) among the labels [97].

HC is also a particular case of Multi-label classification [97], where the labels form a hierarchical structure of Tree or DAG (Direct Acyclic Graph) type. For instance, consider Figure 5.1, where each label is represented as a node of the tree. For a set of labels $\Omega_Y = \{\lambda_1, \dots, \lambda_i, \dots, \lambda_j, \dots, \lambda_L\}$ belonging to a hierarchy, where each λ_i and λ_j are possible labels of class Y, if we represent the relation is-descendent-from as $\prec$, the labels should fulfill the following properties:

- **Asymmetry**: If $\lambda_i \prec \lambda_j$, then $\lambda_j \nprec \lambda_i$, $\forall \lambda_i, \lambda_j \in \Omega_Y$
- **Anti-reflexivity**: $\lambda_i \nprec \lambda_i, \forall \lambda_i \in \Omega_Y$
- **Transitivity**: If $\lambda_i \prec \lambda_j$ and $\lambda_j \prec \lambda_k$, then $\lambda_i \prec \lambda_k, \forall \lambda_i, \lambda_j, \lambda_k \in \Omega_Y$

A HC classifier h assigns each instance $\boldsymbol{x} \in \Omega_{X_1} \times \cdots \times \Omega_{X_M} \subset \mathbb{R}^M$ a vector $\boldsymbol{y}$ of class values:

$$\begin{aligned} h : X &\to P_Y \\ \boldsymbol{x} &\mapsto h(\boldsymbol{x}) = \boldsymbol{y} \end{aligned} \tag{5.1}$$

where P_Y is a set of subsets of Ω_Y, and if $\lambda_i \in h(\boldsymbol{x})$, and the function $ancestors(\lambda_i)$ returns the set of ancestor labels from λ_i to the root node, then

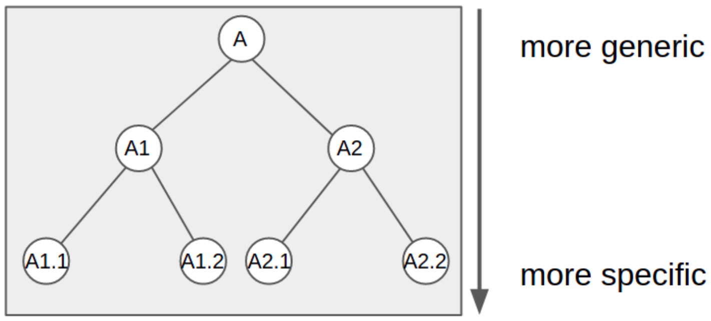


Fig. 5.1: Example of HC label structure, where nodes closer to the root node represent more generic characteristics, while the nodes which are closer to the terminal, also known as leaf nodes, represent more specific characteristics

$ancestors(\lambda_i) \in h(\boldsymbol{x})$, which can be understood as if a particular label λ_i is part of the output of $h(\boldsymbol{x})$, then all the ancestors of λ_i in the hierarchy are also part of the output of $h(\boldsymbol{x})$.

These properties are indeed present on every HC label structure, but there are other characteristics present in HC problems. In Silla and Freitas [97], the following framework is defined in order to describe not only the characteristics of HC problems, but also the characteristics of the algorithms used to solve them. A HC problem is described as a 3-tuple $(\boldsymbol{\Upsilon}, \boldsymbol{\Psi}, \boldsymbol{\Phi})$, where:

- Υ specifies the type of graph representing the hierarchical labels (nodes in the graph) and their interdependencies (edges in the graph). The possible values for this attribute are: T (Tree) or D (DAG-Directed Acyclic Graph).
- Ψ: indicates whether a data instance is allowed to have labels associated with a single path or multiple paths in the class hierarchy. This attribute can take on two values: SPL (Single Path of Labels) or MPL (Multiple Path of Labels).
- Φ: describes the label depth of the data instances: FD (Full Depth Labeling) or PD (Partial Depth Labeling).

An algorithm designed to solve a HC problem can be described as a 4-tuple $(\boldsymbol{\Delta}, \boldsymbol{\Xi}, \boldsymbol{\Omega}, \boldsymbol{\Theta})$, where:

- Δ: indicates whether or not the algorithm can predict labels in just one or multiple (more than one) different paths in the hierarchy. This attribute can take on two values: SPP (Single Path Prediction) or MPP (Multiple Path Prediction).
- Ξ: is the prediction depth of the algorithm. It can have two values: MLNP (Mandatory Leaf-Node Prediction) or NMLNP (Non-Mandatory Leaf-Node Prediction).
- Ω: is the taxonomy structure the algorithm can handle. It has two values: T (Tree) or D (DAG-Directed Acyclic Graph).

- Θ: is the categorization of the algorithm under the proposed taxonomy:
  - LCN (Local Classifier per Node): consists of training one binary classifier for each node of the hierarchy (excluding the root node). The instances labeled with the particular label of the node are considered positive samples, while negative samples can be selected following different strategies (one possibility being 1-vs-all).
  - LCL (Local Classifier per Level): creates a model for each level of the hierarchy. It is the least used strategy in the literature.
  - LCPN (Local Classifier per Parent Node): creates a multiclass classifier for each node that has child nodes, using only instances labeled with labels belonging to the child nodes.
  - GC (Global Classifier): a model that learns the whole class hierarchy, and makes a prediction for all nodes at once.

6 Weakly Supervised Hierarchical Classification

WSC is an umbrella term encompassing various classification problems in which the focus is on learning from incomplete, inexact, and inaccurate supervision, in contrast to supervised classification. A supervised classification problem [60] is formally described by a set of m predictive variables $X = (X_1, \cdots, X_M)$ and a class variable C. Each predictive variable X_i can take a value from its own set of possible values Ω_{X_i} and an instance is a vector $\boldsymbol{x} \in \Omega_{X_1} \times \cdots \times \Omega_{X_M} \subset \mathbb{R}^M$. Specifically, the set of values that the class variable can take, a.k.a. class labels, forms the label space of class variable C. Assuming the existence of an unknown target function $G : \Omega_X \rightarrow \boldsymbol{C}$ that individually categorizes each instance with a single label, the objective of supervised classification techniques, is to build a classifier $\hat{G}$ that approximates the real function G from a set of fully labeled instances $\{(\boldsymbol{x}^1, c^1), \cdots, (\boldsymbol{x}^n, c^n)\}$ of the problem.

WSC problems involve samples that are not described by instance-label pairs, and considering the characteristics of the instance-label relationship highlights the distinctions from other standard supervised classification problems. Solutions proposed for learning from various types of partially labeled data have given rise to the field of WSC.

6.1 Weakly Supervised Hierarchical Classification

Generally, HC problems in the literature predominantly fall into the category of full supervision. Nevertheless, there are instances where alternative supervision models come into play, as exemplified by Santos and Canuto [89]. In their work, they tackle semi-supervised classification utilizing two datasets pertaining to gene functions in a fungus commonly employed in sugar fermentation for ethanol production. Another distinct supervision model is explored by Xiao et al. [123], where they develop a hierarchical text classification model trained on a dataset with noisy labels.

In this paper, we address three WHC scenarios based on three WSC subcategories presented below.

6.1.1 Candidate labels

In a problem of learning from Candidate Labels (CL) [19], instances are provided with a set of possible labels, which includes the true label. A classifier $\hat{W}$ aims to approximate the real function W based on a set of weakly labeled instances, denoted as $\{(\boldsymbol{x^1}, Lx^1), \cdots, (\boldsymbol{x^i}, Lx^i), \cdots, (\boldsymbol{x^n}, Lx^n)\}$. Here, $Lx^i \subseteq \boldsymbol{C}$ represents a set of candidate labels associated with instance $\boldsymbol{x^i}$. It is assumed that Lx^i includes the true label c^i, and an instance $\boldsymbol{x^i}$ is considered ambiguous if $|Lx^i| > 1$. It's worth noting that while this definition encompasses scenarios like fully supervised ($|Lx^i| = 1$) and unsupervised ($|Lx^i| = L$), we typically refer to CL when $1 < |Lx^i| < L$. When the labels of the class to predict are organized hierarchically, we denote this problem as WHC-CL.

6.1.2 Label proportions

In the context of Learning from Label Proportions (LLP) [37], instances are presented without individual labels and are instead grouped into mutually exclusive bags, with each instance belonging exclusively to one bag. Although the labels for these instances are known, the specific pairing of labels to instances has been lost for some reason. Consequently, each bag comprises two unpaired groups of equal size, a group of instances and a group of labels. The group of labels can be presented as the proportion of instances that belong to each class label. Note that these label proportions do not indicate a belief (probability) in the number of instances that belong to each class but the real exact number.

The dataset D of a LLP problem is composed of n unlabeled instances $\{\boldsymbol{x^1}, \ldots, \boldsymbol{x^n}\}$. The instances are provided grouped into b bags, $D = B^1 \cup B^2 \cup \cdots \cup B^b$ where $B^i \cap B^j = \emptyset, \forall i \neq j$. Each bag B^i groups n^i instances, where $\sum_{i=1}^{b} n^i = n$, and n^{ij} denotes the number of instances in B^i which has the label c^j. These n^{ij} values, called counts of the bag B^i, sum up to n^i; i.e., $\sum_{j=1}^{L} n^{ij} = n^i$. Similarly, bag class information can be provided in terms of proportions, $P^{ij} = n^{ij}/n^i \in [0,1]$ with $\sum_{j=1}^{L} P^{ij} = 1$. When the labels of the class to predict are organized hierarchically, we denote this problem as WHC-LLP.

6.1.3 Mutual label constraints

In a problem of learning from Mutual Label Constraints (MLC) [45], the instances are provided unlabeled, but some information in the form of constraints between labels is given. Similarly to the WHC-LLP problem, instances are grouped into bags, and for a given bag, we do know that (i) all labels are different, or (ii) all labels are the same. Therefore, although the true label corresponding to each instance is not known, for each given bag there is a limited number of possible label assignments.

The dataset D of a MLC problem is composed of n unlabeled instances $\{\boldsymbol{x^1}, \ldots, \boldsymbol{x^n}\}$. The instances are provided grouped into b bags $\boldsymbol{B} = \{B^1, \ldots, B^b\}$, where $D = B^1 \cup B^2 \cup \cdots \cup B^b$ and $B^i \cap B^j = \emptyset, \forall i \neq j$. Each bag B^i groups n^i instances, where $\sum_{i=1}^{b} n^i = n$. Let Δ be the property that determines if all the elements of a group have the same label, or different labels, for any element of B^i. We use the function $\Delta : \boldsymbol{B} \rightarrow \{False, True\}$ that maps each bag $B^i \in \boldsymbol{B}$ to either $False$ or $True$ based on whether the labels of the group are all the same or different, respectively. When the labels of the class to predict are organized hierarchically, we denote this problem as WHC-MLC.

6.2 Training with a Top-down learning approach

The defining characteristic of WSC is the presence of uncertainty in label assignment. For example, in the WHC-CL framework, as the number of candidate labels increases, uncertainty correspondingly grows. Similarly, in the WHC-LLP and WHC-MLC scenarios, larger bag sizes result in higher levels of uncertainty.

As expected, the WHC scenarios presented are also subject to this uncertainty. However, employing a Top-down learning approach, inspired by the hierarchical approach introduced by Koller and Sahami [43], can help mitigate this uncertainty.

The top-down learning approach starts by constructing a new dataset, departing from the original dataset, that contains weak label information for the labels associated with the child nodes of the root node. This dataset is then utilized to train a classifier for the root node, positioned in the first tier of the hierarchy. Upon completing the classifier's training, it infers a label for each instance based on the acquired knowledge. Transitioning to the second tier, a similar procedure is executed for each parent node within this tier. For every parent node in the tier, a dataset is generated containing weak label information for the labels corresponding to the child nodes of that parent node. However, this dataset exclusively includes instances that received labels in the preceding tier corresponding to the current parent node-associated label. This process continues for each parent node in each subsequent tier until every instance possesses a complete hierarchical set of inferred labels, extending from the root to the leaf nodes.

This approach results in varying dataset sizes for different parent nodes. Specifically, the training dataset size for parent nodes closer to the leaf nodes tends to be smaller compared to those closer to the root node. However, this imposes the challenge of effective processing of weak information while maintaining its validity across different tiers and dataset sizes.

Figure 6.1 illustrates the application of this approach to a WSC problem based on the hierarchical structure illustrated in Figure 5.1. In the first step, we create a dataset for the root node, where weak information for labels A1 and A2 has to be deduced from the weak information of the leaf nodes. Then in the second step, a classifier is created for the parent node A, which belongs to the first tier of the hierarchy and is also known as the root node. This classifier learns from the dataset created in the first step, containing weak information about labels A1 and A2. Once the model is trained, inference is performed on the dataset, assigning labels to each instance. In the third step, a classifier is created for the parent node A1, the first parent node of the second tier of the hierarchy. This classifier learns from a dataset that contains only instances labeled with the A1 label by the model from the previous tier and performs inference on this filtered dataset. The fourth step involves creating a classifier for the parent node A2, which is the last parent node of the second tier of the hierarchy. Similar to the third step, it learns from a filtered dataset, but in this case, the dataset contains instances labeled with the A2 label by the model from the previous tier. It also performs inference on this filtered dataset, resulting in a fully labeled dataset at the end of the top-down strategy. The pseudocode for our Top-down learning approach is outlined in Algorithm 3.

Algorithm 3 Top-down learning approach for Weakly Supervised Hierarchical Classification

```
Require: H: hierarchy, D: set of training instances with labels information
Ensure: M: a set of models, one for each parent node in the hierarchy
  M ← {∅}
  for $tier$ in $H$ do                    ▷ Iterating from more generic to more specific
      for $parentNode$ in $tier$ do
          $filteredDataset \leftarrow filter\_Data(D, parentNode)$
          $model \leftarrow train\_Model(filteredDataset)$
          $M.append(model)$
          $D \leftarrow infere\_Labels(model, filteredDataset)$
      end for
  end for
  return M
```

The Top-down learning approach offers an advantage in handling large and complex classification problems by decomposing them into smaller and more manageable subproblems. The pseudocode of this approach is elaborated in Algorithm 3, and can applied on any of the scenarios presented. However, the $filter\_Data$ function implementation varies depending on the supervision model used. For the three formally described supervision models, the corresponding $filter\_Data$ function is described in the following sections.

images/TopDownScheme_b.png

Fig. 6.1: Illustration of the top-down strategy, applied to the hierarchy in Figure 5.1. The process unfolds in four steps. First a dataset is created for the parent node, deducing weak label information for labels A1 and A2, from the weak information of the labels corresponding to the leaf nodes. Next, a classifier is trained for parent node A using the dataset generated in the first step, and inference is performed on this dataset, assigning labels to each instance. In the third step, a dataset is created by filtering instances that are not labeled with the A1 label. Subsequently, a classifier for the parent node A1 is created and trained using this newly filtered dataset to perform inference on the filtered dataset. Finally, we create a classifier for parent node A2, following a similar process to the previous step.

6.2.1 Update weak information in WHC-CL

Instances in WHC-CL are represented as tuples $(\boldsymbol{x^i}, Lx^i)$, where $\boldsymbol{x^i}$ denotes the instance features, and Lx^i contains a set of labels that includes the true label. These candidate labels can be converted into probabilistic labels, assigning an equal probability to each candidate label. Although these candidate labels are initially associated exclusively with leaf nodes, we can calculate the probability of labels associated with internal nodes by considering the hierarchical structure.

For example, suppose an instance has candidate labels $Lx^i = (A1.1, A1.2)$. In this case, it is certain to be labeled with both label A and label $A1$, resulting in a vector of label probabilities ($P(A1)$, $P(A2)$, $P(A1.1)$, $P(A1.2)$, $P(A2.1)$, $P(A2.2)$) with values (1, 0, 12, 12, 0, 0).

The Top-down learning approach entails learning from new datasets that contain information derived from these probabilistic labels associated with child nodes. For instance, considering the hierarchy shown in Figure 5.1, if an instance has candidate labels $Lx^i = (A1.1, A1.2, A2.1)$ with probabilities $(P(A1.1), P(A1.2), P(A2.1)) = (13, 13, 13)$, during the first stage of the Top-down learning approach, we can reduce the labels to $Lx^i = (A1, A2)$ with probabilities $(P(A1), P(A2)) = (23, 13)$, effectively reducing uncertainty.

As we progress through the Top-down learning approach, subsequent tiers refine candidate labels by considering only those that are descendants of the assigned node, creating a new filtered dataset. For instance, if the label assigned to an instance in the first tier of the hierarchy is $A1$, the next tier will restrict the candidate labels to $Lx^i = (A1.1, A1.2)$ with probabilities $(P(A1.1), P(A1.2)) = (12, 12)$. Similarly, if the assigned label is $A2$, the candidate labels are narrowed down to $Lx^i = (A2.1)$ with probability $(P(A2.1)) = (1)$, resulting in a significant reduction in uncertainty.

6.2.2 Update weak information in WHC-LLP

During the initial steps of the Top-down learning approach, label proportions can be dynamically recalibrated, considering only the child nodes stemming from the parent node currently under processing. This streamlined approach significantly reduces the number of potential labels in play. To illustrate, we refer to the hierarchy depicted in Figure 5.1. Imagine we have a bag containing 10 instances with label proportions $(p_{A1.1}, p_{A1.2}, p_{A2.1}, p_{A2.2}) = (0.6, 0.1, 0.2, 0.1)$. At the first stage of the Top-down learning approach, we exclusively focus on labels $A1$ and $A2$, resulting in revised proportions $(p_{A1}, p_{A2}) = (0.7, 0.3)$.

After successfully resolving the first step while adhering to these proportions (i.e., class value is assigned), subsequent steps delve into sub-bags containing instances exclusively labeled with $A1$ or $A2$, and label proportions are recomputed accordingly. These sub-bags are: B_{A1} (with a bag size of 7) and B_{A2} (with a bag size of 3). The proportions for the former would be $(p_{A1.1}, p_{A1.2}) = (0.857, 0.143)$, and for the latter, they would be $(p_{A2.1}, p_{A2.2}) = (0.666, 0.333)$. This cascading process leads to a substantial reduction in both the pool of available labels and bag sizes at each stage of the Top-down learning approach.

6.2.3 Update weak information in WHC-MLC

During the initial step of the Top-down learning approach, similarly to WHC-LLP, labels for each group undergo recalibration with a focus solely on the child nodes originating from the currently processed parent node. This streamlining yields a notably reduced set of potential labels. For example, consider a bag denoted as B^i with a size of 4 and $\Delta(B^i) = False$. In this scenario, the available labels to be assigned, which originally included ($A1.1$, $A1.2$, $A2.1$, $A2.2$), when processing root node A, can be simplified to just $(2 \times A1, 2 \times A2)$. Conversely, if $\Delta(B^i) = True$, the label must be chosen from among ($A1.1$, $A1.2$, $A2.1$, $A2.2$), and this set can be simplified to $(A1, A2)$. This reduction in the number of label choices streamlines decision-making and diminishes uncertainty.

6.3 Experimental framework

We have designed a series of experiments with three primary objectives: (1) to assess the advantages of integrating hierarchical information into the learning process, (2) to evaluate the precision of our proposed strategies, (3) to assess the performance of our proposals under diverse experimental scenarios.

The classification tasks performed in the Top-down learning approach require the completion of missing label information for each instance. In this paper, we employ three distinct implementations of the Expectation Maximization (EM) algorithm [21], each tailored for a specific supervision model. As observed in several other works [23, 37, 99, 122], the utilization of techniques grounded in the EM strategy is common. The approach adopted for these experiments is based on the PMEM method presented in Hernández-González et al. [37] (where P indicates that this is a probabilistic version of the EM, and M refers to a Markov Chain Monte Carlo (MCMC) procedure used to obtain an approximate probabilistic completion in the data-completion stage of the EM).

In our experiments, we adhere to the values for the parameters proposed in the work of Hernandez et al. (2013) [37]. For the EM approach, which will learn Bayesian network classifiers (naive Bayes), we employ a threshold to ascertain parametric convergence. This threshold halts the iterative process when the relative difference between the maximum likelihood estimates of two consecutive models falls below 0.1%, as the default setting. Additionally, the maximum number of iterations is fixed at 200. The MCMC-based versions of our methods necessitate two additional parameter values: 1,000 samples for the burn-in phase and 10,000 samples to approximate the label probability expectation.

To evaluate the advantages of integrating hierarchical information into the learning process, we also conduct training on a flat version of the HC scenarios employing the same algorithms. The flat strategy disregards the hierarchical structure and exclusively considers the leaf nodes of the hierarchy. Specifically, in the case of WHC-MLC, we undertake a reassignment of the predicted labels to eliminate any ambiguity in the label name assignment, given that the model lacks access to this hierarchical information during training. This comparative analysis enables us to assess the performance of both hierarchical and non-hierarchical approaches across the three synthetic scenarios, thereby elucidating the added value of incorporating hierarchical information.

6.3.1 Synthetic scenarios

To evaluate the effectiveness of the proposed strategies on WHC datasets, we conduct simulations based on the supervision models we have described. We follow the methodology outlined in Montenegro et al. [62] to generate HC scenarios. These HC scenarios are then modified to fit the weakly supervised settings, aligning them with each of the three formalized supervision models we have introduced.

6.3.1.1 HC synthetic scenarios

Hierarchical datasets in the real world, such as the TieredImageNet Dataset [83], often feature intermediate nodes that share similarities with their sibling nodes. Figure 7.7 provides a visual representation of the hierarchical label structure within this dataset. Here, musical instruments are descendants of a parent node labeled "Instruments" since they all share the common attribute of producing music. Furthermore, intermediate nodes can group instruments based on their shared characteristics, such as whether they belong to the categories of string, wind, percussion, or electric instruments.

To create synthetic datasets that faithfully capture the characteristics of real hierarchical datasets, it is essential to generate instances that encompass not only the attributes of a leaf node but also those of its ancestor nodes. Consequently, an instance generated through this process comprises a concatenation of attributes associated with all the nodes linking the root node to the specific leaf node to which the instance belongs. The root node itself is excluded from this concatenation since it is a universal attribute common to all leaf nodes, offering no discriminatory power. Figure 7.8 visually illustrates this principle, representing attributes as colored shapes, and showcasing instances formed by concatenating these attributes.

The instance generation process is outlined in Algorithm 5. For a given leaf node, this algorithm iterates through the nodes that form the path from the root node of the hierarchy to the leaf node. The set of nodes constituting this

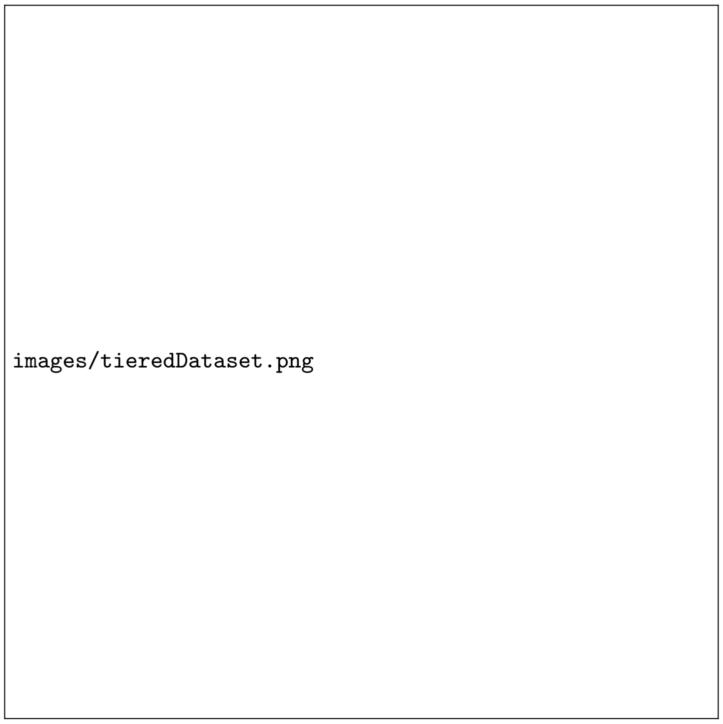


Fig. 6.2: A simplified hierarchical structure of the labels of the TieredImageNet dataset illustrated in [83]. The image shows some of the nodes that are part of the hierarchy as an illustrative example of the scope of the dataset.

path is obtained through the function *genPath(LN)*, where *LN* represents the specific leaf node. During this iteration, features are sampled from each node along the path, and these sampled features are concatenated to create the instance. To perform the sampling, the algorithm relies on the function *genRandomVariable(node)*, which draws random samples from the distribution assigned to the respective node.

Algorithm 4 Generation of a HC instance

```
Require: LN: leaf node for the instance to be created, DS: distributions assigned
  to each node (internal and leaf nodes)
Ensure: instance
  instance ← {∅}
  for node in genPath(LN) do
      instance.concatenate(genRandomVariables(DS[node]))
  end for
  return instance
```

`images/InstanceComposition.png`

Fig. 6.3: Illustration of the method followed to generate synthetic scenarios for a HC problem. Each coloured symbol represents the properties that an instance must have to be labeled with a particular label. Note that, being a hierarchy, instances labeled with a label associated with a terminal node must have not only properties of the leaf label, but also all the ancestors (Extracted from Montenegro et al. [62]).

In our experiments, we utilize the *make_classification*[1] function provided by *scikit-learn* [Guyon] to generate the required distributions for each node in the hierarchies. This function enables us to sample from clusters of normally distributed points (std=1) around the vertices of an n-dimensional hypercube (n refers to the number of features). By adjusting the Class separation (*classep*) parameter, it becomes possible to modify the length of the sides of the hypercube, thereby varying the complexity of the classification task. Importantly, the *Class separation* parameter maintains the same value across all the distributions assigned to each node and dimension within each experimental scenario. This allows us to create scenarios where all classification problems exhibit uniform complexity, while also enabling us to generate scenarios with varying levels of overall complexity across all classification problems.

The conducted experiments incorporate the default value for the *classep* parameter (*classep*=1.0) within the make_classification function. In addition, the hierarchies are defined with depths of 4 and 6, resulting in a total of 8 and 32 leaf nodes, respectively. Within the WHC-CL and WHC-MLC scenarios, the number of candidate labels spans from 2 to the maximum possible value, determined by the total number of leaf nodes present in each hierarchy.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Furthermore, the WHC-LLP scenarios encompass varying bag sizes, ranging from 2 to 100.

Each parameter configuration generates a HC dataset containing 150 instances per class. Subsequently, the dataset is divided into train and test sets, with a test proportion of 33%. The train set is then transformed into the three WHC scenarios presented in this study. To ensure the robustness of the results, this entire process is repeated 10 times, and the reported results represent the average performance across these 10 repetitions.

6.3.1.2 WHC-CL setup

To simulate a WHC-CL dataset, we begin by creating a label set for each instance that initially contains only the true label. Following this, we augment each label set by randomly selecting additional leaf node labels. The probability of adding a label is determined in proportion to its hierarchical distance from the true label. This approach prioritizes candidates that share more characteristics with the true labels, thus creating a labeling scenario that better mimics real-world conditions.

6.3.1.3 WHC-LLP setup

The WHC-LLP setup is generated by assembling bags of the desired size of randomly selected instances. These bags are then used to determine label proportions. The instance labels within each bag are replaced with the corresponding bag label proportions, representing the relative frequency of each label type found within the bag.

6.3.1.4 WHC-MLC setup

To construct each bag on WHC-MLC, two parameters are required: bag size and $\Delta(B)$. When $\Delta(B) = True$, a label is randomly chosen, and instances with that label are sampled at random until the target bag size is reached. When $\Delta(B) = False$, the bag is formed by randomly selecting instances with a label not yet included in the bag. This process continues until the desired bag size is attained or no additional instances from the desired label are available. Note that, when $\Delta(B) = False$, the bag size is constrained by the total number of leaf nodes in the hierarchy since each instance in the bag must have a distinct label.

6.4 Results and discussion

The results obtained from the experiments are analyzed separately for each supervision model used in the study. A thorough examination of the performance of each model is presented, allowing us to gain valuable insights. Based on these individual findings, we draw a comprehensive conclusion that underscores the overall effectiveness of the proposed strategy in comparison to the flat strategy.

6.4.1 Results for WHC-CL

Figures 6.4 and 6.5 display the results of the experiments conducted. Each figure presents the results for a set of experiments using different depth parameter values. Each figure comprises two graphs. The graph on the left shows the average accuracy and the standard deviation in a scenario when the number of extra candidate labels increases, while the graph on the right displays the average computational time required for these experiments.

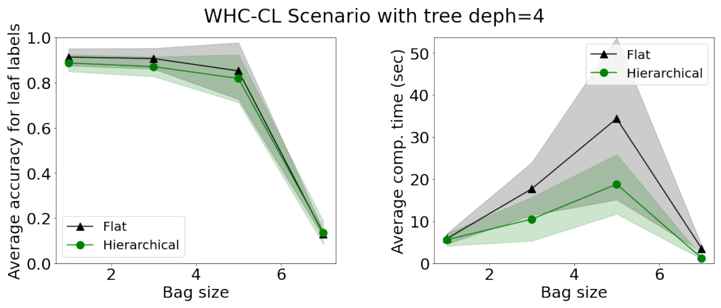


Fig. 6.4: Performance comparison of the hierarchical and flat strategies on WHC-CL scenarios with a depth of 4. The X-axis represents the number of additional candidate labels added to each instance, while the Y-axis indicates the corresponding accuracy and computational time.

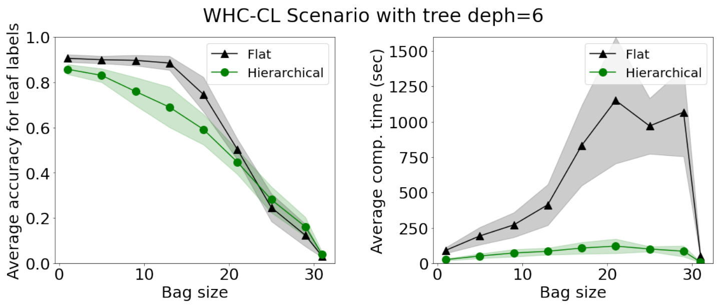


Fig. 6.5: Performance comparison of the hierarchical and flat strategies on WHC-CL scenarios with a depth of 6. The X-axis displays the number of additional candidate labels added to each instance, while the Y-axis represents the corresponding accuracy and computational time. For improved clarity, it is recommended to view the figure in color.

An analysis of Figures 6.4 and 6.5 reveals that as the number of candidate labels increases, both strategies experience a decline in performance due to the introduction of more uncertainty. While the performance of the two strategies is comparable at a depth of 4, at a depth of 6 we can find small differences on performance depending on the total number of candidate labels per instance. However, the computational time advantage of the hierarchical strategy increases as the hierarchy depth grows, significantly favoring the hierarchical approach in terms of computational efficiency.

Therefore, the hierarchical strategy emerges as a notably efficient and scalable approach, particularly in scenarios with a substantial number of labels where training the flat strategy might be impractical. While it may yield slightly lower accuracy in certain specific scenarios, the hierarchical approach presents a distinct advantage in terms of computational efficiency, especially when confronted with extensive hierarchies.

6.4.2 Results for WHC-LLP

Similarly to the previous section, Figures 6.6 and 6.7 show some of the results of the experiments performed for WHC-LLP scenarios.

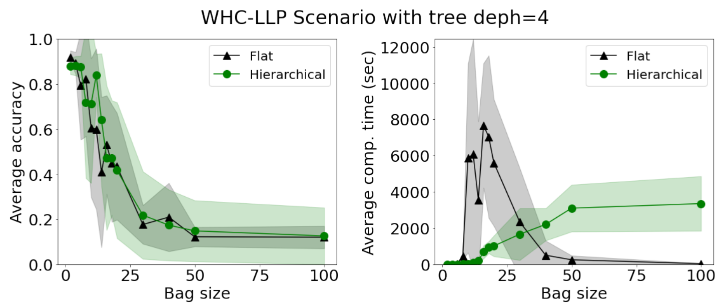


Fig. 6.6: Performance comparison of the hierarchical and flat strategies on WHC-LLP scenarios with a depth of 4. The *bag size* is shown on the X axes, while the Y axes represent the corresponding accuracy and computational time.

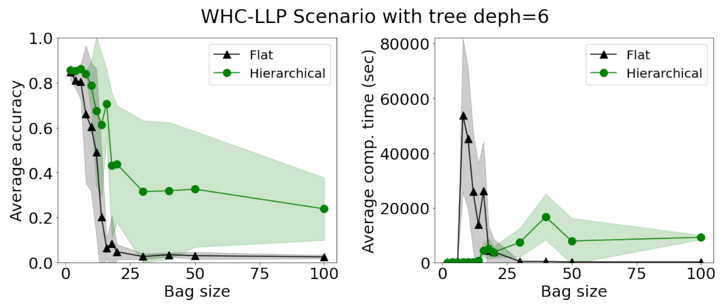


Fig. 6.7: Performance comparison of the hierarchical and flat strategies in WHC-LLP scenarios with a depth of 6. The X-axis represents the bag size, while the Y-axis corresponds to the accuracy and computational time. For a better viewing experience, it is recommended to view the figure in color.

An analysis of Figures 6.6 and 6.7 suggests that the performance of both strategies deteriorates as the number of elements per group increases, resulting in increased uncertainty in the predictions. While both strategies perform similarly at a depth of 4, the hierarchical strategy outperforms the flat strategy as the depth increases. However, it is essential to note that the computational time advantage of the hierarchical approach for small bags is reversed for larger bags. In these cases, slightly more computational time is necessary, but the hierarchical strategy yields significantly superior results, whereas the flat strategy produces notably poor performance results.

It is expected that the hierarchical strategy will require more computing time for scenarios with larger bags. As the total number of instances remains fixed for all scenarios within a given hierarchy, an increase in bag size leads to a decrease in the total number of bags, affecting each strategy differently. In the worst-case scenario, a flat strategy performs an MCMC process for each existing bag. In contrast, a hierarchical strategy, in the worst-case scenario, performs a number of MCMC processes equal to the total number of internal nodes in the hierarchy ($internal\_nodes = 2^{d-1}$ for a hierarchy with depth d and 2 children per node). While the hierarchical strategy benefits from faster convergence of MCMC processes due to the simplifications outlined in Section 6.2.2, it is important to note that as the bag size increases, there is a higher likelihood of requiring a greater number of MCMC processes. Additionally, these processes may require extra time to reach convergence. The presence of an upper limit on the total number of iterations for the MCMC process, which consequently imposes a cap on the allocated computing time for each MCMC process, results in a scenario where the hierarchical strategy, depending on the bag size, may require more computing time compared to the flat strategy. This occurs because the hierarchical strategy resolves fewer MCMC processes that reach the predefined upper limit.

Despite the increased computing time associated with the hierarchical strategy in scenarios with larger bag sizes, it remains substantially lower than the maximum computing time required by the flat strategy in scenarios with smaller bags. This remarkable difference highlights the computational efficiency of the hierarchical strategy, particularly when compared to the computationally demanding nature of the flat strategy in scenarios characterized by small bag sizes.

6.4.3 Results for WHC-MLC

Similarly to the previous sections, Figures 6.8 and 6.9 show some of the results of the experiments performed for WHC-MLC scenarios.

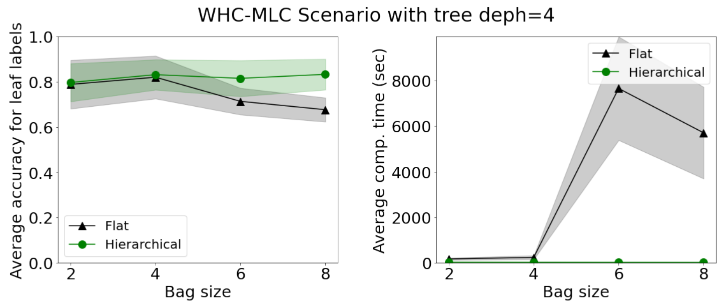


Fig. 6.8: Performance comparison of the hierarchical and flat strategies on WHC-MLC scenarios with a depth of 4. The *bag size* is shown on the X axes, while the Y axes represent the corresponding accuracy and computational time.

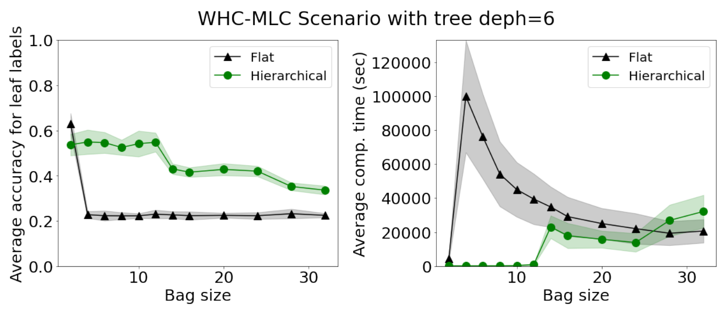


Fig. 6.9: Performance comparison of the hierarchical and flat strategies on WHC-MLC scenarios with a depth of 6. The *bag size* is shown on the X axes, while the Y axes represent the corresponding accuracy and computational time.

An analysis of Figures 6.8 and 6.9 suggests that the performance of both strategies slightly deteriorates as the number of elements per group increases. The hierarchical strategy exhibits comparable or slightly superior performance compared to the flat strategy across varying hierarchy depths. This observed trend appears to stem from the inherent capability of the hierarchical strategy to mitigate the complexity of each bag, eliminating the necessity for computationally intensive MCMC processes, which do not guarantee the optimal solution. In scenarios where there is a significant difference in computational time, it is likely that fewer MCMC processes are being executed, which could lead to errors. However, when the computational time gap is reduced, it can be assumed that a similar number of MCMC processes are being calculated, thereby inducing comparable errors, as illustrated in Figure 6.8 for bag sizes of 2 and 4, and in Figure 6.9 for bag sizes between 14 and 32.

However, it is worth noting that the difference in computing time between the Flat and Hierarchical strategies becomes more pronounced as the depth of the hierarchy increases. This is expected because the number of instances is related to the depth of the hierarchy, leading to more bags and potentially more MCMC processes in the hierarchical strategy.

Nonetheless, as the bag size increases, this difference in computation time reduces due to having a smaller number of bags, and the hierarchical strategy cannot avoid using the MCMC process.

An analysis of WHC-MLC scenarios containing only one type of bag (similar labels or bags with different labels) reveals that, despite time improvements in both cases, the most significant time improvement comes from solving bags with different labels (as shown in Figures 6.10 and 6.11). It is essential to emphasize that both scenarios collectively contribute to the performance advantage favoring the hierarchical strategy.

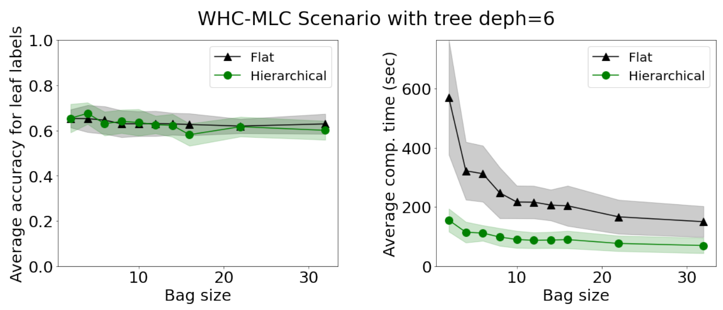


Fig. 6.10: Performance comparison of the hierarchical and flat strategies on WHC-MLC scenarios with a depth of 6 and bags with similar labels. The *bag size* is shown on the X axes, while the Y axes represent the corresponding accuracy and computational time.

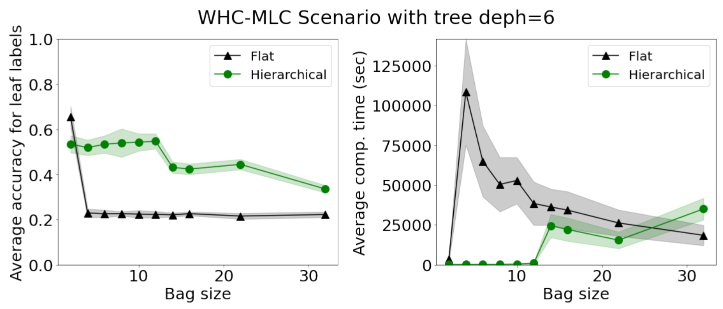


Fig. 6.11: Performance comparison of the hierarchical and flat strategies on WHC-MLC scenarios with a depth of 6 and bags with different labels. The *bag size* is shown on the X axes, while the Y axes represent the corresponding accuracy and computational time.

6.5 Conclusions and future work

In this study, we have introduced and examined three distinct WHC scenarios. We have also proposed a general strategy that incorporates hierarchical knowledge during the training phase to address the classification tasks associated with these scenarios. Based on our experimental findings, we conclude that incorporating hierarchical information into the learning process leads to enhancements in classifier performance. In summary, our experiments indicate that integrating hierarchical information into the learning process can enhance the performance of multilabel classification tasks.

Some of the insights from our analysis include the following:

- The hierarchical strategy consistently showed superior or comparable performance compared to the flat strategy across most scenarios, even when dealing with large hierarchical structures.
- While our hierarchical strategies demonstrated computational efficiency in WHC-CL and WHC-MLC scenarios, the impact on WHC-LLP scenarios was less conclusive, mainly influenced by the bag size.
- Overall, the hierarchical strategy effectively reduced computing time, with the benefits being particularly pronounced in scenarios featuring extensive label hierarchies.

Furthermore, while we have introduced hierarchical strategies that enhance performance and computing time efficiency, we recognize that further advancements are possible. It is important to note that our primary focus in this work was to verify the benefits of considering hierarchical information in comparison to the flat strategy. As a result, we did not specifically address scenarios involving labels that are not leaf nodes.

For instance, one could explore scenarios in which WHC-CL includes candidate labels that are not necessarily leaf nodes, or WHC-LLP scenarios in which bags consist of proportions of labels from the same depth of the hierarchy but not necessarily the leaf nodes. Similarly, in the context of WHC-MLC, scenarios could be designed with bags containing similar labels up to a certain depth, with variations beyond that point.

While introducing flat strategies into such scenarios might create an unfair comparison due to the inherent advantages of hierarchical information, our demonstrated benefits underscore the potential for further exploration. Future work could involve investigating these variations in other scenarios and comparing different hierarchy-based strategies to gain deeper insights.

Multi-Dimensional Hierarchical Classification

7.1 Introduction

7.1.1 Multi-Dimensional Classification

MDC is a supervised learning paradigm in which multiple class variables (dimensions) $\boldsymbol{Y} = (Y_1, \ldots, Y_d, \ldots, Y_D)$ need to be jointly predicted [82]. An MDC classifier h assigns each instance $\boldsymbol{x} \in \Omega_{X_1} \times \cdots \times \Omega_{X_M} \subset \mathbb{R}^M$ a vector $\boldsymbol{y}$ of class values:

$$\begin{aligned} h : \boldsymbol{x} &\to \Omega_{Y_1} \times \cdots \times \Omega_{Y_D} \\ \boldsymbol{x} &\mapsto h(\boldsymbol{x}) \end{aligned} \tag{7.1}$$

We assume that Y_d is a discrete class variable, for all $d \in \{1, \ldots, D\}$ where Ω_{Y_d} denotes its sample space and $|\Omega_{Y_d}|$ is the cardinality of the sample space of Y_d. Given the expression in Equation 7.1, we can formulate other traditional classification paradigms as particular cases of MDC:

- $D = 1$ and $|\Omega_{Y_1}| = 2$: Binary classification
- $D = 1$ and $|\Omega_{Y_1}| > 2$: Multi-class classification
- $D > 1$ and $|\Omega_{Y_d}| = 2$, $\forall d$: Multi-label classification

7.1.2 Multi-dimensional dependencies

The main feature distinguishing multi-dimensional classification from a regular classification task is that a number of class variables has to be predicted simultaneously [103]. Thus, it is obviously important to exploit potential dependencies between class labels to obtain the best performance on classification. As mentioned previously, in this work two types of label dependence are distinguished, namely *conditional* and *unconditional dependence*.

We can consider *conditional dependence* between two binary class variables Y_i and Y_j given x when:

$$p(Y_i, Y_j|\boldsymbol{x}) \neq p(Y_i|\boldsymbol{x})p(Y_j|\boldsymbol{x}) \tag{7.2}$$

which intuitively represents the dependence between two class variables given some characteristic of the predictive variables. There is *unconditional dependence* between two binary class variables Y_i and Y_j when:

$$p(Y_i, Y_j) \neq p(Y_i)p(Y_j) \tag{7.3}$$

which intuitively represents the dependence between two classes, irrespective of the characteristics of the predictive variables.

Unconditional dependence can be easily tested using the labels of a dataset and Equation 7.3, however, testing *conditional dependence* can be a complex task which involves learning from the predictive variables. While *conditional* and *unconditional dependencies* have a strong connection [20] and can be useful to improve the performance of MDC classifiers, ultimately it is *conditional dependence* which is more relevant to classification performance, since that is where the information from the predictive variables is considered [82].

MDC strategies have been proven to outperform *Single-task learning* (STL) strategies, especially when there are only a few training examples per task and the tasks are related [82]. In the literature, multiple approaches take advantage of these two types of dependencies to improve the performance of the classifiers trained for a particular MDC problem. One common strategy is to train a classifier to jointly predict dependent class variables to obtain better performance. In order to identify which class variables are dependent, Tenenboim-Chekina et al. [103] present a method that uses the chi-square (X^2) score to measure *unconditional dependencies.* Tenenboim-Chekina et al. [103] also propose a method to identify dependent classes based on information obtained from *conditional dependence* using a strategy where, for a given pair of classes, the performance of training two independent classifiers is compared to the performance of an MDC classifier. An improvement in the performance of the MDC classifier gives an indirect measure of the *conditional dependence* between the two classes, since the model has taken into account the characteristics of the features during training.

Hernandez-Leal et al. [38] have also used the concept of *unconditional dependence* to detect dependencies between classes, but, in this case, the classes are split into two groups: independent classes, for which a classifier for each class is trained separately, and dependent classes for which a single classifier is trained to predict all dependent classes at once. This method however, does not use *conditional dependence* information to differentiate the classes.

Other approaches, such as those proposed by Zhang and Zhang [127] and Read et al. [82], consider an efficient way of measuring pairwise conditional dependence in MDC data, based on the fact that maximising the likelihood of the data is equivalent to minimising the mutual information between the data instances and the error. They consider the binary classification problem as a special case of the nonlinear regression problem:

$$y = f(\boldsymbol{x}) + e \tag{7.4}$$

where y denotes the target variable, $\boldsymbol{x}$ the set of predictors, e the noise and f is a smooth function. Given the examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, fitting the above model with maximum likelihood is equivalent to minimizing the mutual information between x and the estimate of e. The proof of this proposition can be found in the Appendix of Zhang and Zhang [127].

A more detailed review of dependence (in the MDC context) and the strategies used in the literature can be found in Dembczynski et al. [20].

7.1.3 Multi-Dimensional Hierarchical classification

MDHC is a supervised learning paradigm in which multiple class variables (dimensions) $\boldsymbol{Y} = (Y_1, \ldots, Y_d, \ldots, Y_D)$ need to be jointly predicted. Each dimension d has a vector of labels $\boldsymbol{\Omega_{Y_d}} = (\lambda_1^d, \ldots, \lambda_i^d, \ldots, \lambda_j^d, \ldots, \lambda_{L_d}^d)$ forming a hierarchical structure, where $L_d > 0$ and each λ_i^d and λ_j^d are possible labels of the class variable Y_d. Similarly to HC, the labels of a dimension can be represented as nodes of a tree structure as illustrated in Figure 5.1. If we represent the relation is-descendent-from as $\prec$, the labels of each dimension should fulfill the following properties:

- **Asymmetry**: If $\lambda_i^d \prec \lambda_j^d$, then $\lambda_j^d \not\prec \lambda_i^d$, $\forall \lambda_i^d, \lambda_j^d \in \boldsymbol{\Omega_{Y_d}}$
- **Anti-reflexivity**: $\lambda_i^d \not\prec \lambda_i^d, \forall \lambda_i^d \in \boldsymbol{\Omega_{Y_d}}$
- **Transitivity**: If $\lambda_i^d \prec \lambda_j^d$ and $\lambda_j^d \prec \lambda_k^d$, then $\lambda_i^d \prec \lambda_k^d, \forall \lambda_i^d, \lambda_j^d, \lambda_k^d \in \boldsymbol{\Omega_{Y_d}}$

A MDHC classifier h assigns each instance $\boldsymbol{x} \in \Omega_{X_1} \times \cdots \times \Omega_{X_M} \subset \mathbb{R}^M$ a vector $\boldsymbol{y} = (y_1, \ldots, y_d, \ldots y_D)$ of sets of predicted class labels, where $L_d \geq |y_d| \geq 0$, being $|y_d|$ the cardinality of the predicted label set y_d:

$$\begin{aligned} h : X &\to P_{Y_1} \times \cdots \times P_{Y_D} \\ \boldsymbol{x} &\mapsto h(\boldsymbol{x}) \end{aligned} \tag{7.5}$$

where $\boldsymbol{P_{Y_d}}$ is the set of subsets of $\boldsymbol{\Omega_{Y_d}}$, and if $\lambda_i^d \in h(\boldsymbol{x})$, then $ancestors(\lambda_i^d) \in h(\boldsymbol{x})$, which can be understood as if a particular label λ_i^d is part of the output of $h(\boldsymbol{x})$, then all the ancestors of λ_i^d are also part of the output of $h(\boldsymbol{x})$.

An example of a MDHC problem can be found In Montenegro et al. [61], a dataset with two class variables ("Topic' and '"Intent"), where each class variable contains a hierarchical set of labels, is presented. The unconditional dependencies between the most important labels is briefly analyzed using the illustration shown in Figure 4.4 and the goal of this unconditional dependence analysis is to better understand the dataset without solving the classification task. However, this unconditional dependence study shows that these dependencies exist, and that they can be exploited, but it was beyond of the scope of that work to do so.

Another source of MDHC problems is the result of applying the "Divide and Conquer" strategy to already existing HC problems defined with

(Ψ=MPL). In Peng et al. [72] the RVC1 [48] and NYTimes [88] HC problems are divided into 5 and 9 sub-problems respectively, with the objective of reducing the learning complexity for a GC. However, in this work the problem is not presented as MDHC, and each HC problem is solved independently and later on combined with some restrictions for the cases where the sub-problems are nested.

Naik et al. [65] show a peculiar version of MDHC problems which can be generated combining two different datasets labeled with different hierarchies on the same topic. In this case we can consider this problem a partially-labeled MDHC, since the datasets are independent, and combining them we obtain a dataset where each instance is labeled only on one class variable whilst missing the other class variable label. However, the problem is not presented as MDHC, and two different strategies are proposed to solve the problem, one approach based on Transfer Learning (TL) and the other approach based on a Single-dimensional approach. However, Naik et al. [65] exploit the dependencies between the two HC problems in order to either combine datasets, or transfer already learnt information to another training process. The method used to find those dependencies is based on the proximity of the centroidal vectors of each label, where a centroidal vector is a combination of the vector representations of the instances that each label has.

Domains in which it is more common to find HC problems, and therefore prone to MDHC problems, are text classification ([17], [43] or [59]), protein function prediction ([6], [86] or [117]) or music genre classification ([15], [49] or [58]).

However, none of these works have presented and studied the MDHC paradigm and its characteristics, they have just solved a particular MDHC problem with a different strategy, or presented a MDHC dataset.

7.2 Dependencies in MDHC

In HC, the inherent properties of hierarchical structures defined in Chapter 5 imply unconditional dependencies from each child node to its respective parent node, since every instance that belongs to a particular node also belongs to the ancestor node. In MDHC, these hierarchy dependencies coexist with dependencies between nodes from different dimensions, and therefore different hierarchies. The label of a node u in a hierarchy can be interpreted as a concatenation of labels $u = \{\boldsymbol{t_0^u}, \ldots, \boldsymbol{t_X^u}\}$ where each t_x^u gives information about a node of the unique direct path that connects the root node with the node u. For example, taking Figure 7.1 as reference, the label for node $A2.1$ can be represented as the concatenation of $\{\mathbf{A,A2,A2.1}\}$.

The degree of the *unconditional dependence* between two labels in a MDHC problem, can be measured by analyzing the dataset. This degree of *unconditional dependence* can be defined as $D(u, v)$, being u and v two nodes of any of the hierarchies in the MDHC problem. Multiple methods can be used to

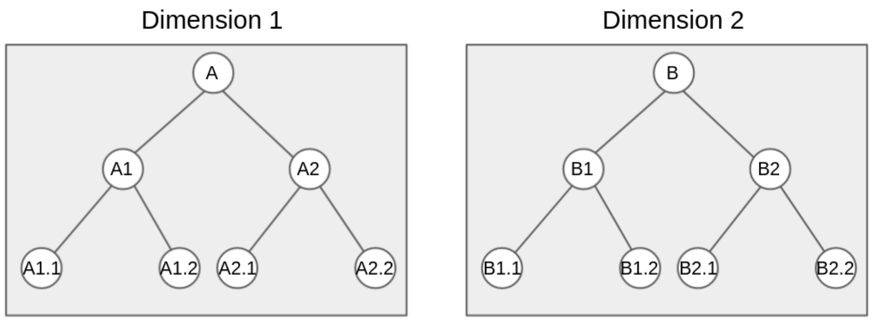


Fig. 7.1: Example of MDHC label structure. The tree structure on the left represents the HC label set of one dimension, while the tree structure on the right belongs to a second dimension. The nodes of each tree represent the available labels for each dimension.

measure *unconditional dependence* such as the Chi-square test [103], correlation coefficients [20], centroidal vectors [65] or number of common instances [61]. In this work we measure the degree of *unconditional dependence* as the conditional probability:

$$D(u,v) = P(v|u) = \frac{|P(v \cap u)|}{|P(u)|}$$

which can be represented as a matrix of dependencies as in other MDC classification problems such as Wang et al. [115], adapted to MDHC.

7.3 MDHC performance measures

We propose the following extension of the performance measures existing in the MDC domain [8] to the MDHC domain, by incorporating the HC performance measures defined by Kiritchenko et al. [42]. We define the auxiliary functions for *multi-dimensional hierarchical precision* (δhP) and *multi-dimensional hierarchical recall* (δhR):

$$\delta hP(d,n) = \frac{|P_{dn} \cap T_{dn}|}{|P_{dn}|}$$

$$\delta hR(d,n) = \frac{|P_{dn} \cap T_{dn}|}{|T_{dn}|}$$

These functions evaluate the predicted labels for a specific test example defined by d (index of the dimension) and n (index of the example). P_{dn} is the set consisting of the most specific predicted label or labels (in the case of *Multiple Path of Labels*) for test example n in dimension d and all its or their ancestor labels, and T_{dn} is the set consisting of the true most specific label(s) of test example n in dimension d and all its(their) ancestor labels. Using these auxiliary functions, we define *mean hierarchical precision* ($\overline{HP}$), *mean hierarchical recall* ($\overline{HR}$) and *mean hierarchical F measure* ($\overline{HF}$) as:

$$\overline{HP} = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{N}\sum_{n=1}^{N}\delta hP(d,n)$$

$$\overline{HR} = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{N}\sum_{n=1}^{N}\delta hR(d,n)$$

$$\overline{HF} = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{N}\sum_{n=1}^{N}\frac{2*\delta hP(d,n)*\delta hR(d,n)}{\delta hP(d,n)+\delta hR(d,n)}$$

where D is the number of dimensions of the problem, and N is the number of instances used to evaluate the performance. Following the same strategy, we define the auxiliary functions for *global hierarchical precision* ($\widehat{hP}$) and *global hierarchical recall* ($\widehat{hR}$):

$$\widehat{hP}(n) = \frac{|\widehat{P}_n \cap \widehat{T}_n|}{|\widehat{P}_n|}$$

$$\widehat{hR}(n) = \frac{|\widehat{P}_n \cap \widehat{T}_n|}{|\widehat{T}_n|}$$

where:

$$\widehat{P}_n = \bigcup_{d=1}^{D} P_{dn}$$

$$\widehat{T}_n = \bigcup_{d=1}^{D} T_{dn}$$

Similarly to $\delta hP(d,n)$ and $\delta hR(d,n)$, $\widehat{hP}(n)$ and $\widehat{hR}(n)$ functions evaluate the predicted labels for a specific test example indexed by n. $\widehat{P}_n$ is the set consisting of the most specific labels predicted for test example n in each dimension and all their ancestor labels, and $\widehat{T}_n$ is the set consisting of the true most specific label(s) of test example n in each dimension d and all their ancestor labels.

We define *global mean hierarchical precision* ($\widehat{HP}$) , *global mean hierarchical Recall* ($\widehat{HR}$) and *global F measure* ($\widehat{HF}$) as:

$$\widehat{HP} = \frac{1}{N}\sum_{n=1}^{N}\widehat{hP}(n)$$

$$\widehat{HR} = \frac{1}{N}\sum_{n=1}^{N}\widehat{hR}(n)$$

$$\widehat{HF} = \frac{1}{N}\sum_{n=1}^{N}\frac{2 * \widehat{hP}(n) * \widehat{hR}(n)}{\widehat{hP}(n) + \widehat{hR}(n)}$$

where N is the number of test instances.

7.4 MDHC Strategies

While MDHC problems can be solved by MDC strategies ignoring the hierarchical characteristics of the problem, and also by HC strategies ignoring the MDC aspect of the problem, these strategies do not benefit from the relationships between labels of the same hierarchy, or from different dimensions. This information has been proven to be beneficial in MDC problems such as the problems described in Bielza et al. [8] and Hernandez-Leal et al. [38], however the benefit of these relationships between labels in MDHC problems had yet to be investigated.

In Section 7.5.1, we will show that it is indeed possible to construct scenarios where dependencies within and across hierarchies arise. In the following, we focus on the classification strategies needed to address these scenarios.

In order to propose specific MDHC classification strategies, we combine two MDC strategies with two HC strategies to generate four different proposals. The MDC strategies have been chosen to represent the two most frequently used types of classification strategies in the literature:

- **Stacking**: replace the original predictions, obtained by learning every label separately, by correcting them in light of information about the predictions of the other labels [121]. This transformation of the initial prediction should be interpreted as a regularization procedure: a bias is introduced, in an attempt to decrease the variance.
- **Grouping**: Based on a correlation coefficient measured between each pair of classes, classes are split up into two groups: independent classes, which are trained using a classifier for each class variable separately, and dependent classes which are trained group-wise by a single classifier [112].

Regarding HC, we also use two of the most common strategies for HC classification [97], namely LCPN (*Local Classifier per Parent Node*) and GC (*Global Classifier*), with the following characteristics: SPP (*Single Path Prediction*), MLNP (*Mandatory Leaf-Node Prediction*) and the taxonomy of the label structure will be T (Tree). Following the framework mentioned in Chapter 5, and based on the MDHC scenarios that are generated with the methodology explained in Section 7.5.1, the two HC strategies can be defined as:

- **LCPN** = ($\mathit{\Delta}$:**SPP**, $\mathit{\Xi}$:**MLNP**, $\mathit{\Omega}$:**T**, $\mathit{\Theta}$:**LCPN**)
- **GC** = ($\mathit{\Delta}$:**SPP**, $\mathit{\Xi}$:**MLNP**, $\mathit{\Omega}$:**T**, $\mathit{\Theta}$:**GC**)

As a result, we present four different MDHC strategies described as:

- **Stacking + LCPN**: This is a straightforward method, where we follow the Stacking strategy, where in a first phase a LCPN strategy is applied for each dimension separately and the input vectors are the feature vectors. In the second phase, we again apply a LCPN strategy to each dimension independently, however, instead of using the feature vectors as input, we use the vector of probabilities resulting from concatenating the predictions of all the classifiers from the first phase. This MDHC strategy is illustrated in Figure 7.2 as if it were applied to the MDHC problem illustrated in Figure 7.1, where each circle represents a classification task at each parent node. A variation of this strategy has been previously used on a HC problem in Hernández et al. [36], where a HC classifier inspired by MDC is proposed, however, the problem itself is not MDC nor MDHC, but an extension of the LCPN strategy.

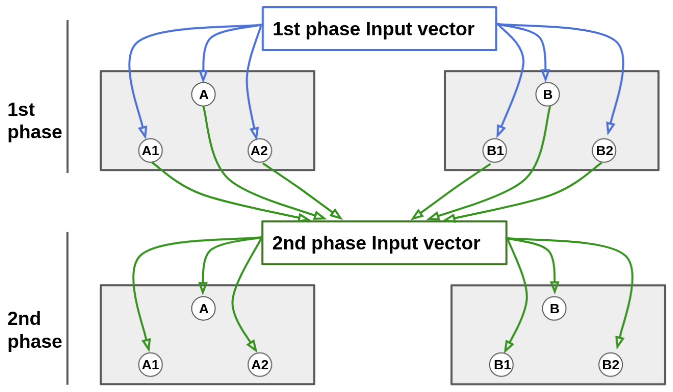


Fig. 7.2: Illustration of the Stacking+LCPN strategy. The feature vector is used on the 1^{st} phase as input, while the 2^{nd} phase uses the outputs of the 1^{st} phase. At each phase, based on the LCPN strategy, for each dimension a set of classifiers for each parent node is created.

- **Stacking + GC**: Similarly to the previous strategy, in the first phase a GC strategy is applied to each dimension separately using the feature vectors as input, and the second phase again applies a GC for each dimension separately but with the vector of probabilities resulting from concatenating the predictions of the classifiers of the first phase as input. This MDHC strategy is illustrated in Figure 7.3 as if applied to the MDHC problem illustrated in Figure 7.1, where each circle represents a classifier which solves the classification problem of the parent nodes listed inside.

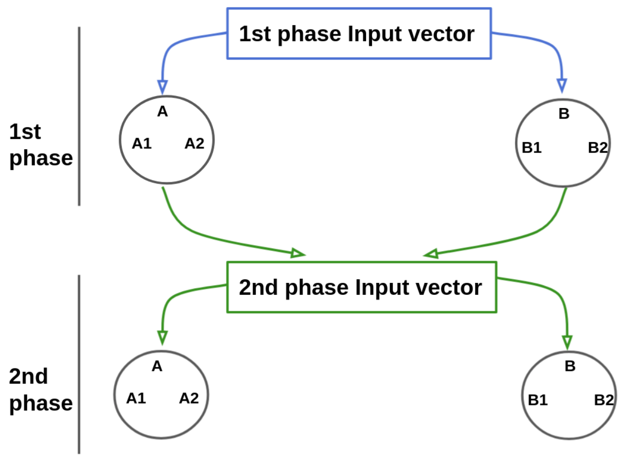


Fig. 7.3: Illustration of the Stacking+GC strategy. The feature vector is used on the 1^{st} phase as input, while the 2^{nd} phase uses the outputs of the 1^{st} phase. At each phase, a GC classifier for each dimension is created, and it is trained to solve all the classification tasks associated with the internal nodes drawn inside the circle.

internal

- **Grouping + LCPN**: In this strategy, based on a dependency measure and the classification tasks resulting from applying the LCPN strategy to each dimension, related classification tasks are grouped and solved together. This MDHC strategy is illustrated in Figure 7.4 as if applied to the MDHC problem illustrated in Figure 7.1, and supposing that the nodes are related as illustrated in Figure 7.5.
- **Grouping + GC**: In this case, the GC strategy would create a GC classifier for each dimension separately, a grouping version of this strategy groups all dimensions into one GC classifier. Therefore, we train a single GC classifier that has as output all the labels for both hierarchies, as illustrated in Figure 7.6, where each circle represents a classifier which solves the tasks listed inside.

For a particular instance, these strategies estimate the probability of belonging to each node of the MDHC problem. In order to decide what labels should be assigned, the class-prediction top-down approach proposed by Koller and Sahami [43] is followed for each hierarchy independently, selecting the labels in a top-down fashion, as follows. For each level of the hierarchy (except the top level), the decision about which label is assigned at the current level is based on the label predicted at the previous (parent) level. For example, following the hierarchies from Figure 7.1, suppose the estimated probability for node $A1$ is greater than the probability estimated for $A2$. At the next level, only

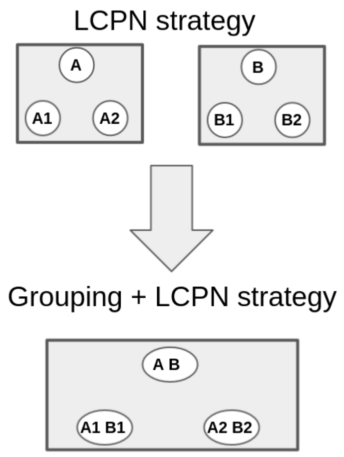


Fig. 7.4: Illustration of how a LCPN strategy is transformed into a Grouping+LCPN strategy. Instead of using a LCPN strategy for each dimension, the classification tasks of each LCPN problem are grouped according to a criteria, and then solved as a single problem.

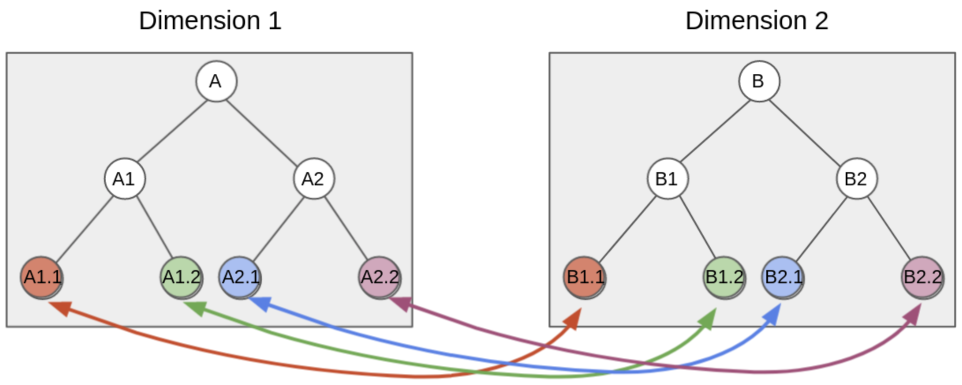


Fig. 7.5: Illustration of dependencies between labels of the synthetic MDHC hierarchies. Having all dimensions the same structure, each node has a degree of dependency with the nodes on the same position of the hierarchy of other dimensions. This dependency is represented as an arrow connecting the nodes from different dimensions.

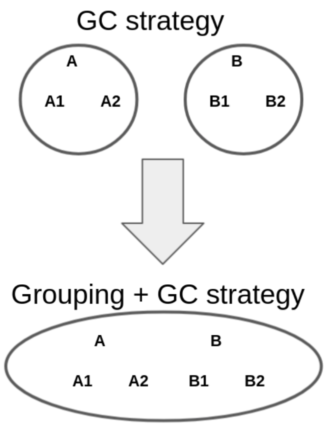


Fig. 7.6: Illustration of how a GC strategy is transformed into a Grouping+GC strategy. Instead of using a GC strategy for each dimension, a GC strategy is applied to all the dimensions simultaneously. A GC strategy is designed to solve multiple classification tasks simultaneously, in this image each GC is represented as a circle, and all those tasks are drawn inside.

the children nodes of node A1 are considered. This strategy guarantees that the predicted labels fulfill the HC properties described in Section 5.1.

7.5 Synthetic scenarios

In order to evaluate and compare MDHC strategies, it is required to develop an experimentation where the strategies are evaluated on multiple and variate MDHC scenarios. However, due to the novelty of this paradigm, the datasets are scarce and we cannot rely exclusively on real datasets. Therefore, in this section we present a procedure to generate MDHC synthetic scenarios that will allow us to overcome the scarcity.

Synthetic datasets have been used in multiple Machine Learning areas of research such as MDC [54], HC classification [95], ML classification [105], Weak-label classification [85] or Semantic segmentation of images [74] among others. However, none of these match the requirements for MDHC, therefore we present a procedure to generate MDHC synthetic datasets.

Serrano-Pérez and Sucar [95] presented a procedure to generate HC datasets where, for a given hierarchy, a distribution is assigned to each leaf node and, in order to generate instances for the nodes, values are sampled from each distribution. This process is followed to generate instances with paths that finish in a leaf node, however, in order to generate instances with paths that finish in an internal node, for each internal node a normal distribution is estimated using samples of all its leaf nodes descendants, and finally instances are sampled from that distribution. Although this is an easy to understand and implement method, the hierarchies generated with this approach could be understood as the result of applying a "Divide and Conquer" strategy to a Multi-class (MC) classification problem. The internal nodes generated by this process are groupings by proximity of the classes of a MC problem, and they lack a real characteristic that relates them.

Some real hierarchical datasets such as "TieredImageNet Dataset" [83] show how grouped nodes share characteristics with all sibling nodes on the hierarchical representation. A simplified representation of the hierarchy of this dataset can be found in Figure 7.7, where, as an example, musical instruments are descendant nodes from a parent node named "Instruments", since they are all devices created or adapted to make musical sounds. Other intermediate nodes that could be added to this hierarchy could group instruments depending on their characteristics based on whether they are string, wind, percussion or electric instruments for example.

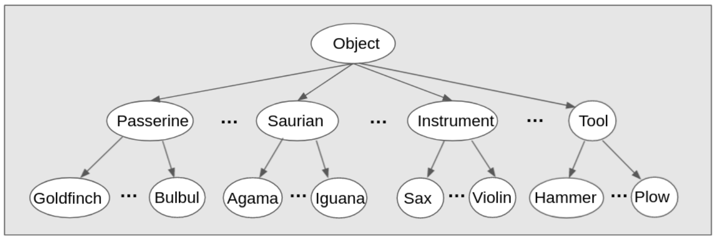


Fig. 7.7: A simplified hierarchical structure of the labels of the TieredImageNet dataset illustrated in [83]. The image shows some of the nodes that are part of the hierarchy as an illustrative example of the scope of the dataset.

In order to create synthetic datasets that mimic the characteristics of real hierarchical datasets, we must create instances that contain not only characteristics of a leaf node, but also characteristics of internal nodes. Therefore, an instance generated by this process is formed by the concatenation of characteristics from all the nodes that link the root node and the leaf node where the instance belongs (excluding the root node, since it is common to all leaf nodes, and therefore useless in terms of classification). This concept is illustrated in Figure 7.8, where characteristics are represented as coloured shapes, and instances are a result of the concatenation of characteristics.

Algorithm 5 shows how, for a given leaf node, we can iterate over the nodes linking the root node of a hierarchy to the leaf node (*genPath(n)* returns the set of nodes that reach leaf node n from the root node), and generate an instance by concatenating the features sampled from each node (the function *genRandomVariable(node)* samples from the distribution assigned to *node*).

Algorithm 5 Generation of an HC instance

```
Require: leafNode: leaf node for the instance to be created, distributions: distributions assigned to each node (internal and leaf nodes)
Ensure: instance
  instance ← []
  for node in genPath(leafNode) do
      instance.concatenate(genRandomVariables(distributions[node]))
  end for
  return instance
```

We can extend Algorithm 5 to generate MDHC instances as shown in Algorithm 6. The input in this case is composed of multiple leaf nodes, one for each dimension, and the output vector is the equivalent of the concatenation of the HC instances of each dimension.

Algorithm 6 Generation of a MDHC instance

```
Require: multipleLeafNodes: list of leaf nodes for each dimension of the instance to be created, distributions: distributions assigned to each node (internal and leaf nodes of all dimensions)
Ensure: instance
  instance ← []
  for leafNode in multipleLeafNodes do
      for node in genPath(leafNode) do
          instance.concatenate(genRandomVariables(distributions[node]))
      end for
  end for
  return instance
```

This process is illustrated in Figure 7.8, where we can see the features (represented with a coloured figure) that belong to each particular instance, which determines to what nodes the different instances belong to. This method is done in the same way as in the other dimensions, and in order to generate MDHC instances with information from every dimension, these features are concatenated. In order to generate specific degrees of dependency between nodes, the adequate number of instances with each label combination must be selected to achieve the desired proportions.

In these experiments we use the *scikit-learn* auxiliary function *make_classification*[1] to create the distributions needed for each node of each hierarchy. This function permits us to sample from clusters of points normally distributed (std=1) about vertices of an n-informative-dimensional hypercube. The sides of this hypercube can be reduced in terms of length by modifying the *Class separation* parameter, making the classification problem more difficult. Note that

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

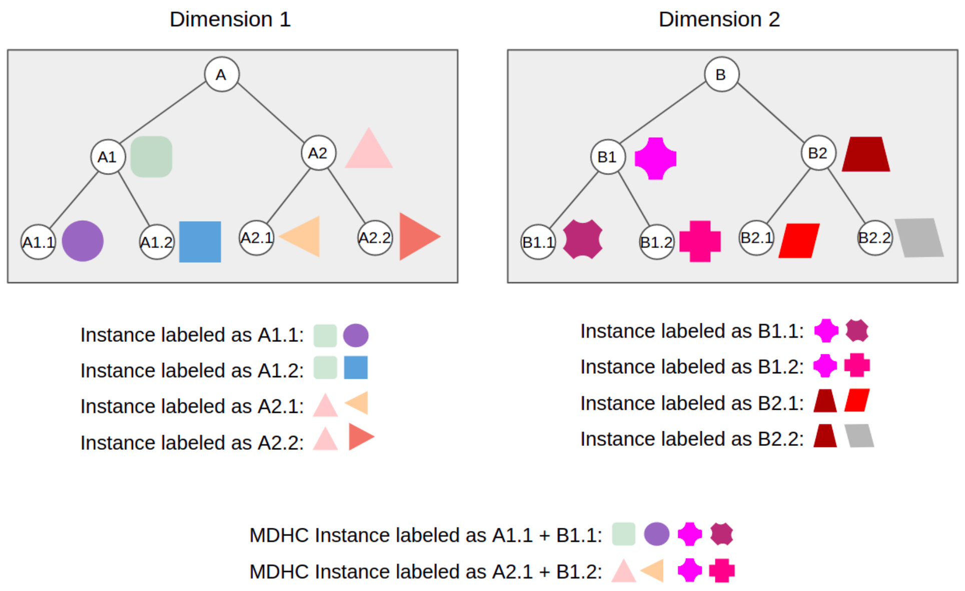

Dimension 1
Dimension 2
A
A1
A2
A1.1
A1.2
A2.1
A2.2
B
B1
B2
B1.1
B1.2
B2.1
B2.2
Instance labeled as A1.1:
Instance labeled as A1.2:
Instance labeled as A2.1:
Instance labeled as A2.2:
Instance labeled as B1.1:
Instance labeled as B1.2:
Instance labeled as B2.1:
Instance labeled as B2.2:
MDHC Instance labeled as A1.1 + B1.1:
MDHC Instance labeled as A2.1 + B1.2:

Fig. 7.8: Illustration of the method followed to generate synthetic scenarios for a MDHC problem with 2 dimensions. Each coloured symbol represents the properties that an instance must have to be labeled with a particular label. Note that, being a hierarchy, instances labeled with a label associated with a terminal node must have not only properties of the terminal label, but also all the ancestors.

for each scenario, the *Class separation* remains constant for all the distributions assigned to each node for all dimensions. We will therefore use the *Class separation* parameter to generate scenarios with more complicated classification problems, instead of manually overlapping the distributions as seen in Serrano-Pérez and Sucar [95].

7.5.1 Synthetic scenarios generated

The datasets for the synthetic scenarios generated in this experimentation are based on a MDHC problem which is composed of D HC problems described as (Υ=T, Ψ=SPL and Φ=FD) according to the framework description from Section 5.1. The selection of these characteristics has been carried out with the objective of creating intuitive classification problems which do not distract from the goals of this work, but are also representative of the hierarchical problems found in the literature. The Υ=T and Ψ=SPL characteristics have been selected because they are more common than their alternatives according to Silla and Freitas [97]. However, Φ=FD has been set looking for simplicity in the experiments performed on synthetic scenarios, although for the other experiment on a real dataset presented in Section 7.6.2 it has Φ=PD due to the nature of that particular MDHC problem.

In order to study how characteristics of MDHC problems can influence the performance of different MDHC classification strategies, we define a set of synthetic scenarios with different possible assignments for the characteristics: *children per node*, *degree of dependency* and *class separation*. For the experimentation, three MDHC scenarios with different values for the *children per parent node* characteristic and two levels are created. The dependencies between nodes will follow the pattern illustrated in Figure 7.5 and the *degree of dependency* between labels will take values from $\{\boldsymbol{Balanced}, \mathbf{0.3}, \mathbf{0.6}, \mathbf{0.9}\}$. The *Balanced* value represents the scenario where there are no strong dependencies between terminal nodes of different dimensions, even so the value is not zero and depends on the structure of the hierarchies. Particularly, if U= {terminal nodes of hierarchy A} and V= {terminal nodes of hierarchy B} then:

$$\forall_{u \in U, v \in V} \Rightarrow D(u,v) = \frac{1}{|V|} \tag{7.6}$$

Using these *degrees of dependency*, the dependency matrices for the MDHC scenarios can be easily calculated. For example, for scenarios with *children per parent node*=2, *Depth=3* and *Dimensions=2*, the dependency matrices take the values shown in Tables 7.1-7.4. Note that these dependency matrices only contain information about the dependencies between terminal nodes, which is enough to determine the dependencies between all labels in scenarios described as (Υ=T, Φ=FD), and these matrices show how related are labels from different dimensions. Nevertheless, these scenarios also need to define dependencies with internal labels, otherwise the rows and columns might not sum up to 1, as they should. Tables 7.1-7.4 show how as the degree of dependency grows, for a given label the dependency with other label from the other dimension grows, whilst the dependencies with the rest of the labels decrease.

	B1.1	**B1.2**	**B2.1**	**B2.2**
A1.1	0.25	0.25	0.25	0.25
A1.2	0.25	0.25	0.25	0.25
A2.1	0.25	0.25	0.25	0.25
A2.2	0.25	0.25	0.25	0.25

Table 7.1: Dependency matrix for a scenario with *children per parent node*=2, *Depth=3*, *Dimensions=2* and *degree of dependency*=Balanced

7.6 Experimental framework

In this section we present the experimental framework. The goal of the experiments is twofold: 1) To evaluate how dependencies between labels and

	B1.1	**B1.2**	**B2.1**	**B2.2**
A1.1	0.300	0.233	0.233	0.233
A1.2	0.233	0.300	0.233	0.233
A2.1	0.233	0.233	0.300	0.233
A2.2	0.233	0.233	0.233	0.300

Table 7.2: Dependency matrix for a scenario with *children per parent node*=2, *Depth=3*, *Dimensions=2* and *degree of dependency*=0.3

	B1.1	**B1.2**	**B2.1**	**B2.2**
A1.1	0.600	0.133	0.133	0.133
A1.2	0.133	0.600	0.133	0.133
A2.1	0.133	0.133	0.600	0.133
A2.2	0.133	0.133	0.133	0.600

Table 7.3: Dependency matrix for a scenario with *children per parent node*=2, *Depth=3*, *Dimensions=2* and *degree of dependency*=0.6

	B1.1	**B1.2**	**B2.1**	**B2.2**
A1.1	0.900	0.033	0.033	0.033
A1.2	0.033	0.900	0.033	0.033
A2.1	0.033	0.033	0.900	0.033
A2.2	0.033	0.033	0.033	0.900

Table 7.4: Dependency matrix for a scenario with *children per parent node*=2, *Depth=3*, *Dimensions=2* and *degree of dependency*=0.9

other characteristics of MDHC problems can affect the performance of different MDHC classification strategies, 2) To determine if the methods introduced in this work can outperform strategies that do not take into account the hierarchical or multidimensional characteristics of the problem.

7.6.1 Experimental setup in synthetic scenarios

We conduct this analysis generating a set of synthetic scenarios with different characteristics. Using the proposed MDHC performance measures, we compare the MDHC strategies described previously with a MDC strategy, and two HC strategies. The MDC strategy used completely ignores the hierarchical characteristics of the problems during training, creating a Multi-label classifier that predicts, simultaneously for a given instance, the probability of having all existing terminal labels from every dimension. For each dimension of the classification problem, the labels assigned to the particular instance are those connecting the root label with the terminal label whose probability is the highest. No corrections are performed after the MDC prediction, therefore some predicted label sets can be incoherent with the hierarchical properties (i.e., if a prediction contains a label but not its ancestors), which will be penalized by the MDHC performance measures. However, the MDC strategies are provided with all the dependency information available in the form of the dataset during training, and therefore it is possible to infer the HC characteristics of the MDHC problem, this being fair to compare with the other strategies. The decision to not include any correction based on the hierarchy, is to avoid considering this MDC strategy as a HC strategy such as GC, or the MDHC strategy Grouping+GC. Regarding the HC strategies, each dimension is evaluated independently using the LCPN and GC strategies, and although the dependencies between labels from different dimensions are ignored, the predicted labels are coherent with the hierarchical properties.

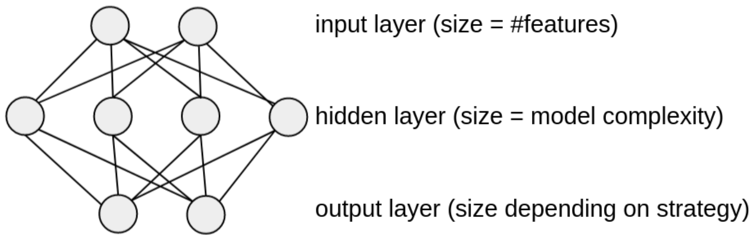

input layer (size = #features)
hidden layer (size = model complexity)
output layer (size depending on strategy)

Fig. 7.9: Architecture of the base NN classifier used on the experimentation with synthetic scenarios.

The implementation of these strategies is done using feed forward neural networks (NN) as the base classifier. This network is formed by one input layer, one hidden layer, and an output layer, where all three are fully connected layers built with Keras[2] as illustrated in Figure 7.9. The parametrization of NN is key to its performance, therefore in these experiments the size of the hidden layer takes a range of values that allows us to avoid underfitting. In order to avoid overfitting, the training dataset of each scenario is divided into train and test sets (train:70% - test:30%), additionally, for the *Stacking*-based strategies, the train set is split into two halves, one for the training process of each part of the stack (train 1^{st} stack:35% - train 2^{nd} stack:35% - test:30%). The 1^{st} part of the stack receives as input the raw features of the 1^{st} subset, whilst the 2^{nd} part of the stack receives as input the distribution of probabilities of the labels (predicted by the 1^{st} part of the stack) for each instance in the 2^{nd} subset.

Therefore the different parameters that will be varied to generate the synthetic scenarios are:

- Class separation: $\{\mathbf{0.01, 0.1, 1}\}$
- Dimensions: $\{\mathbf{2, 3, 4}\}$

[2] https://www.tensorflow.org/guide/keras

- Depth of hierarchies: $\{\mathbf{2, 3, 4}\}$
- Children per parent node: $\{\mathbf{2, 3, 4}\}$
- Degree of dependency between classes: $\{\boldsymbol{Balanced}, \mathbf{0.3, 0.6, 0.9}\}$

Regarding the implementation of the classifiers, the parameters that will be varied are:

- Model complexity (hidden layer size): $\{\mathbf{2^0, 2^1, \ldots, 2^{10}}\}$
- Strategies: {**MDC, LCPN, GC, Stacking + LCPN, Stacking + GC, Grouping + LCPN, Grouping + GC**}

We assume that the *Grouping + LCPN* strategy takes into account all unconditional dependencies (Figure 7.5), and the related nodes are grouped.

For each terminal node, 1.000 instances are created, consequently, scenarios with more children per parent node have more instances to train, but are also more complex classification problems, since the cardinality of the label set is higher.

Due to the random initialization of the parameters in the NN and the random aspect of the generation of the scenarios, we repeat the whole process 10 times where, for each repetition, the scenarios are generated and new models are created and trained. The final performance is the average of the performances obtained for all repetitions.

7.6.2 Experimental setup on a real MDHC problem

In order to complement and support our conclusions, a comparison between the MDHC strategies and HC strategies is made using a real-world dataset [61] from the Empathic project [108]. The reason for using this dataset is that it is the only one available that is naturally created as a MDHC problem, whilst the other datasets mentioned above need transformations or are Weakly-labeled.

Some Machine Learning tasks derived from the development of the Empathic project belong to the Natural Language Understanding module, where a "Topic" label and an "Intent" label have to be assigned to each utterance coming from a user [61]. This "Topic" and "Intent" classification task is a MDHC problem, and the dependencies between labels have been briefly analyzed in Montenegro et al. [61]. However, in the Empathic dataset the hierarchies are not symmetric, there are 55 different labels in the "Topic" class variable with 5 levels of depth, and 34 different labels in the "Intent" class variable across 4 levels of depth. The Empathic dataset contains 11,383 sentences, which is a small number of sentences compared to other text-classification datasets, such as LSHTC [69] (more than 2,400,000 sentences). The combination of the small size of the dataset with the amount of labels and the unbalanced distribution of instances per label make this problem difficult to solve, therefore we have simplified the label hierarchies down to 3 levels of depth on both hierarchies, as illustrated in Figures 7.10 and 7.11.

This classification problem also allows non-terminal nodes as the final label to be predicted (NMLNP: Non-Mandatory Leaf-Node Prediction), therefore

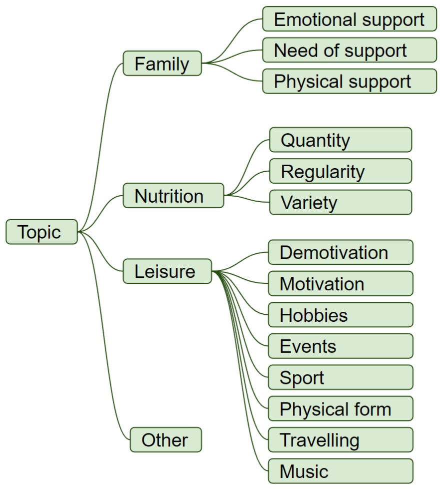


Fig. 7.10: Illustration of the simplified “Topic” tree from the Empathic dataset. Each box represents a node on the tree, and inside each box, the name of the associated label.

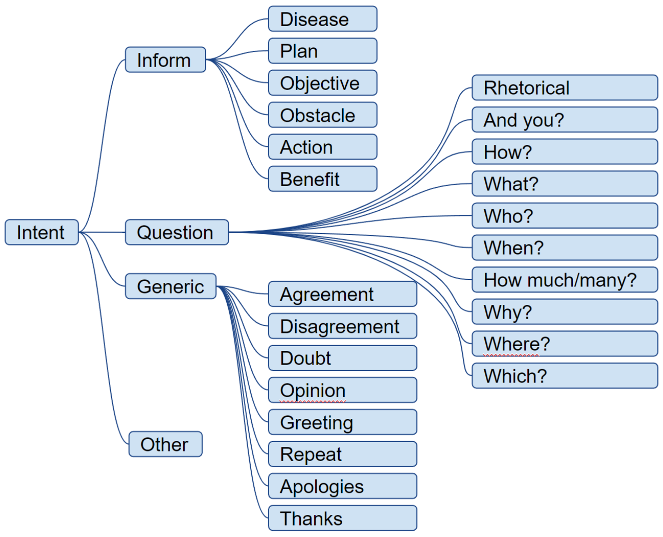


Fig. 7.11: Illustration of the simplified “Intent” tree from the Empathic dataset. Each box represents a node on the tree, and inside each box, the name of the associated label.

following the framework mentioned in Chapter 5 the MDHC strategies can be defined as:

- **LCPN** = (Δ:**SPP**, Ξ:**NMLNP**, Ω:**T**, Θ:**LCPN**)
- **GC** = (Δ:**SPP**, Ξ:**NMLNP**, Ω:**T**, Θ:**GC**)

In this case, the base classifier is a NN trained using the pretrained embeddings from the BERT [22] tokenizer[3], followed by an embedding layer and multiple convolutional layers that generate the feature vector that feeds the final output layer, as illustrated in Figure 7.12.

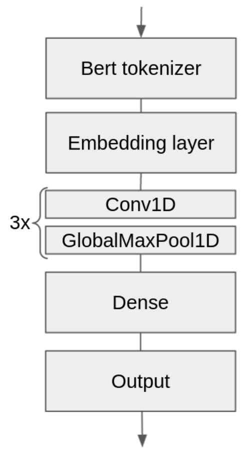


Fig. 7.12: Architecture of the base NN classifier used in the experimentation with the Empathic dataset. Each box represents a node on the tree, and inside each box the name of the associated label.

We have performed the experiments on the Empathic dataset 10 repetitions and averaged the results obtained. The train-test split (80%-20%) are stratified due to the unbalanced distribution of the labels. The rest of the hyperparameters for these experiments can be found in Table 7.5.

[3] https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1

Hyperparameter	Value
Max epochs	100
Embeding dimension	200
conv1D filters	50
conv1D kernel size	3
Dropout rate	0.2
Dense layer activation functions	relu
Dense layer size	128

Table 7.5: Hyperparameters used on the base model used for the Empathic dataset experiments.

Regarding the Grouping+LCPN strategy, the labels have been grouped by pairs selected in descending order according to the metric Chi-Square Test for Labels Dependencies Identification presented by Tenenboim-Chekina et al. [103], applied to the train data. Every group is formed by a parent node from each hierarchy, and each node can only belong to one group, which results in the following groups:

- group1 = "Topic" + "Intent"
- group2 = "Topic_nutrition" + "Intent_inform"
- group3 = "Topic_sportandleisure" + "Intent_generic"
- group4 = "Intent_question"

7.7 Results and discussion

The experiments performed in this work aim at identifying whether MDHC strategies can help to improve the classification performance of MDC and HC strategies on MDHC problems, whose characteristics negatively affect conventional classifiers.

In this section we first analyze the results obtained from the experiments on synthetic scenarios, delving into the effect of each parameter separately. Then, the performances on some specific scenarios is analyzed in order to confirm the premises deduced from the univariate analysis.

Finally, the results of the experiments on the Empathic project dataset are analyzed and compared with the results of the synthetic scenarios.

7.7.1 Synthetic scenarios results

The performance of the strategies evaluated on a particular synthetic scenario can be examined as illustrated in Figure 7.13. This figure is composed of two plots, and each one shows the performances of a set of strategies on a specific scenario, whose characteristics are listed on the title of each plot. Each plot shows the performance measured in terms of Global HF (Y axis) in relation to the number of *neurons in the hidden layer* (X axis) of the strategies noted in the legend. In this work we consider multiple baselines, namely *LCPN*, *GC* and *MDC*, which represent the most common HC and MDC strategies that can be used to solve a MDHC problem.

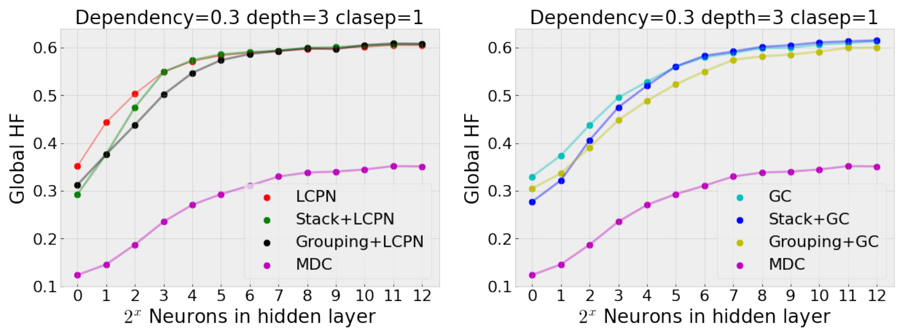


Fig. 7.13: Performance of the MDHC strategies on scenarios with *degree of dependency*=Balanced, *children per parent node*=2 and *class separation*=0.1. The plot on the left shows the performance of the LCPN-based and MDC strategies, and the plot on the right shows the GC-based and MDC strategies. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the number of neurons in the hidden layer of the base models of each strategy. Preferably viewed in color.

In particular, the plot on the left of Figure 7.13 shows the performance of the *LCPN*, *Stacking+LCPN*, *Grouping+LCPN* and *MDC* strategies on the scenario defined by *dependency=0*, *children per parent node=2* and *class separation=0.1*. The different lines show how the performances improve as the models grow in *complexity*, however, they reach a performance upper limit around 2^8 neurons. At this point, despite the fact that increasing the complexity of the models does not result in an improvement in the performance, the decision for using a MDHC strategy can have a significant impact on the final performance of the classifier. This particular scenario shows how the MDC strategy proposed, which ignores all hierarchical information, performs worse in general terms, although it tends to converge in performance with the rest of the strategies as the complexity grows. For low complexity classifiers, HC strategies tend to perform better than any other strategy, however, they are slightly surpassed by the *Stacking+LCPN* strategy as soon as they reach a certain complexity around 2^2. However, *Stacking+LCPN* and *LCPN* show similar values for the whole range of *neurons in the hidden layer*. The plot on the right of Figure 7.13 shows a similar behaviour with the *GC*-based strategies, and, in fact, this is a common occurrence. This behaviour is due to the bottleneck that a small number of *neurons in the hidden layer* creates on the neural network structure of all the strategies proposed. This bottleneck effect has been previously studied in Gupta et al. [31], where the critical bottleneck dimensionality (below which structural information is lost) of autoencoders for text representations is studied. For a MDHC problem such as the one in Figure 7.5, applying a LCPN strategy creates a classifier for each parent node, where the input vectors contain 4 features as input, and 2 outputs (*children per parent node*=2). However, with a Stacking+LCPN strategy, although the first phase has the same architecture as LCPN (as illustrated in Figure 7.2), in the second phase the input vector for each classifier is formed by the concatenation of the outputs of each classifier of the first phase, that is 12 features as input and 2 outputs, creating a more severe bottleneck when a small number of *neurons in the hidden layer* is used. This downside of the Stacking-based strategies also occurs on Grouping-based strategies, since classification problems from different dimensions are combined, and therefore their input vectors are also concatenated, forcing more information to pass through a bottleneck than on a more traditional strategy, such as GC or LCPN strategies (Figures 7.4 and 7.6).

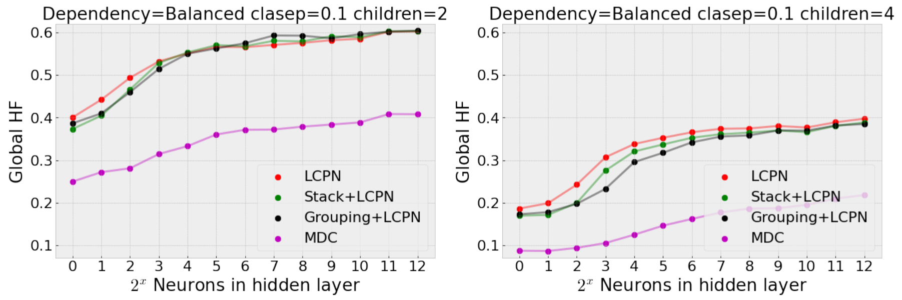


Fig. 7.14: Performance of LCPN-based strategies on scenarios with *degree of dependency*=Balanced and *class separation*=0.1 but varying *children per parent node* with 2 and 8 values. The plot on the left illustrates the results on the scenario with *children per parent node* = 2 and the plot on the right the scenario with *children per parent node* = 8. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the number of neurons in the hidden layer of the base models of each strategy.Preferably viewed in color.

In order to measure how each individual characteristic of the scenarios affects the performance of the different strategies, we perform a two-step univariate analysis. The first step consists of analyzing the average performance of each strategy on all the scenarios for each given value of the characteristic studied. We use a plot to visualize how the strategies are affected by the different values that a particular characteristic can take.

The second step also consists of a univariate analysis of the comparison between two strategies, analyzing the difference of Global HF between each MDHC strategy and its HC counterpart. The usage of this performance measure is due to the variability of performance mainly depending on the number of *children per parent node*, as illustrated in Figure 7.14, where the performance in two different scenarios is shown, and the only difference between the scenarios is the number of *children per parent node.* Averaging the performance in scenarios with different characteristics can show biased results towards strategies that perform better in scenarios with fewer *children per parent node.* Therefore, in order to reduce this bias, in the second step of the analysis the LCPN-based strategies are compared with the performance of the LCPN strategy, and GC-based strategies are compared with the performance of the GC strategy, calculating their difference in performance as:

$$DiffP(s_1, s_2) = GlobalHF_{s_1} - GlobalHF_{s_2} \tag{7.7}$$

7.7.2 Effect of *children per parent node*

Regarding *children per parent node*, Figure 7.15 shows how every strategy is negatively affected as the parameter value grows. The effect on the LCPN-based strategies is smaller than on the other strategies, meaning that the performance of these strategies is less affected by scenarios where the *children per parent node* is higher than the other strategies. The second step analysis, Figure 7.16, does not provide any other insight on what strategies are more suitable than others. From the two-step univariate analysis, we can conclude that, although the *children per parent node* severely affects to all strategies (as seen in Figures 7.14 and 7.15), the LCPN-based strategies can help to mitigate this effect.

7.7.3 Effect of *degree of dependency*

In terms of *degree of dependency* between labels, Figure 7.17 shows how all strategies perform better as the dependency between labels grows. Although this statement can seem counter-intuitive in the case of strategies that are not MDHC, it is an expected behaviour since a scenario with high label dependency means that more features can help to determine the label of an instance. In fact, a MDHC scenario with *degree of dependency*=1 can be reduced to a HC problem when the hierarchies are identical and the dependencies are node to node, since the hierarchical information contributed by the two hierarchies is identical.

However, for *degree of dependency* = Balanced, the performance of every strategy appears to improve. This effect does not occur for values of *children per parent node* = 2, where the performance continues to decrease as the *degree of dependency* shrinks. However, for values of *children per parent node* of 4 and 8, we hypothesize that this effect has to do with the interference of a bias effect

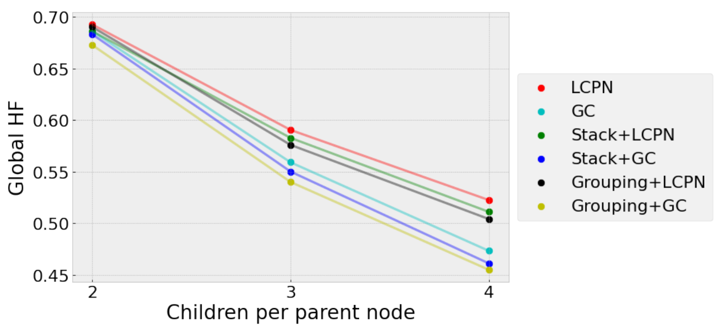


Fig. 7.15: Evolution of the Global HF with respect to the *children per parent node*. The points of each strategy are the result of averaging the performance on all the scenarios that have the *children per parent node* parameter equal to the corresponding value on the X axis. The Y axis shows the measure used for comparison (GlobalHf), and the X axis shows the *children per parent node*.Preferably viewed in color.

caused by the unbalance of the dependencies between labels. This unbalance makes the classifiers biased towards the most populated dependencies. For the sake of clarification, a scenario with *degree of dependency* = 0.3 and *children per parent node* = 2 is created with the following dependencies:

- D(A1.1,B1.1)= 0.3
- D(A1.1,B1.2)= $\frac{0.7}{3}$
- D(A1.1,B2.1)= $\frac{0.7}{3}$
- ...

While a scenario with *degree of dependency* = 0.3 and *children per parent node* = 4 is created with the following dependencies:

- D(A1.1,B1.1)= 0.3
- D(A1.1,B1.2)= $\frac{0.7}{15}$
- D(A1.1,B1.3)= $\frac{0.7}{15}$
- ...

When the *degree of dependency* = Balanced, the dependencies are balanced and the classifiers are not biased, achieving better performance. On the other hand, as the *degree of dependency* increases, the MDHC problem gradually transforms to a single HC problem, with double the number of features available to classify the instances.

Nevertheless, the second step of the analysis (Figure 7.18) shows how, with the exception of the Stacking+LCPN strategy, all MDHC strategies achieve minor performance improvements from a *degree of dependency* increase with respect to their HC counterpart. However, the benefit is more remarkable in the Grouping strategies as illustrated in Figure 7.18.

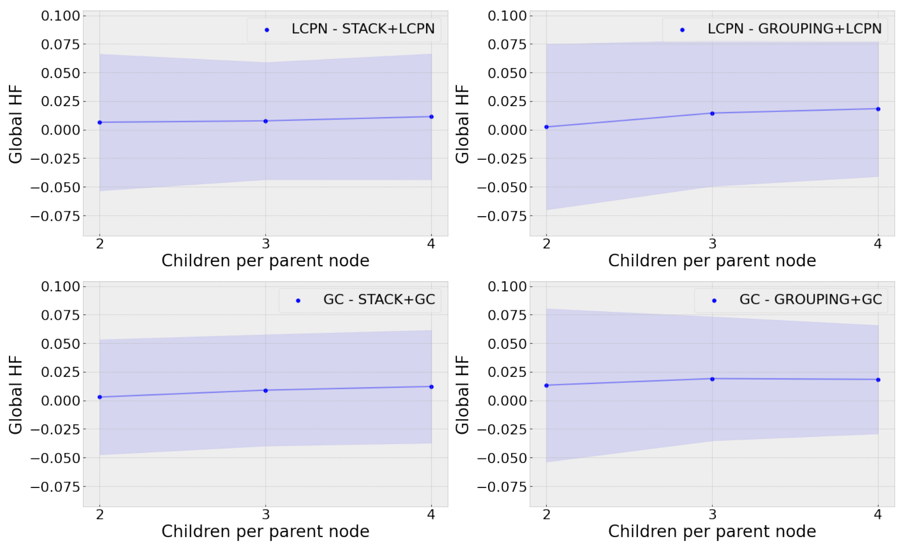


Fig. 7.16: Evolution of the average Global HF difference between a pair of strategies as the number of *children per parent node* grows. Each point is calculated as the average of the differences of the performance of a pair of strategies on all the scenarios that have the *children per parent node* parameter equal to the corresponding value on the X axis. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the *children per parent node*.

7.7.4 Effect of *class separation*

The first step analysis of the *class separation* influence on the MDHC strategies performance from Figure 7.19 shows how a higher degree of *class separation* (easier classification problem) makes non-MDHC strategies increase their performance more significantly the MDHC strategies. What can also be understood as non-MDHC strategies are more penalized by smaller values of *class separation* (more difficult classification problems). The second step analysis reveals how this effect is true for every MDHC version, although the difference between GC and GROUPING+GC is lesser compared to the other MDHC strategies.

7.7.5 Effect of *depth*

The first step analysis of the *depth* shows how the LCPN-based strategies are less affected by the growth of this parameter (Figure 7.21). This was to be expected since *depth* growth implies that the hierarchies contain more internal nodes, and that means more classifiers in LCPN-based strategies, however, other methods do not grow accordingly and need to increase the complexity

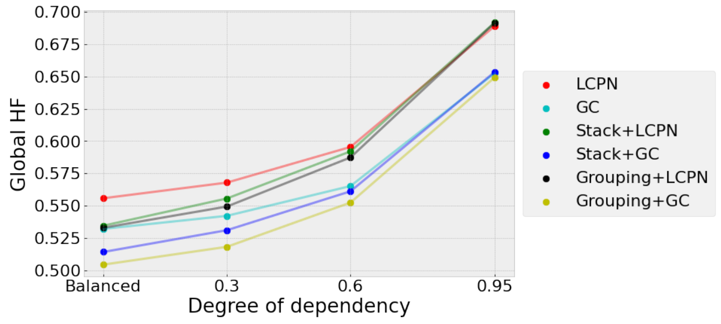


Fig. 7.17: Evolution of the Global HF with respect to the *degree of dependency*. The points of each strategy are the result of averaging the performance on all the scenarios that have the *degree of dependency* parameter equal to the corresponding value on the X axis. The Y axis shows the measure used for comparison (GlobalHf), and the X axis shows the *degree of dependency*.Preferably viewed in color.

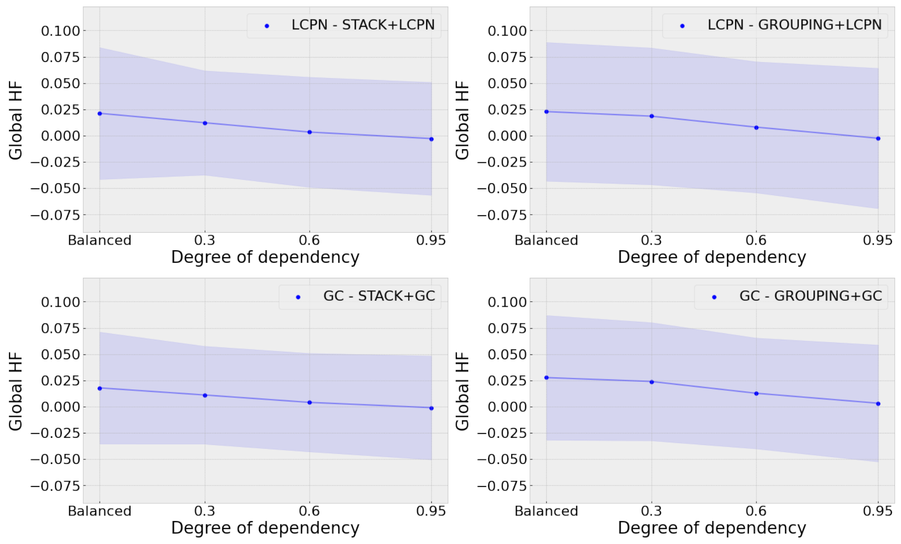


Fig. 7.18: Evolution of the average Global HF difference between the MDHC strategies and their HC counterpart as the *degree of dependency* parameter grows. Each point is calculated as the average of the differences of the performance of a pair of strategies on all the scenarios that have the *degree of dependency* parameter equal to the corresponding value on the X axis. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the *degree of dependency*.

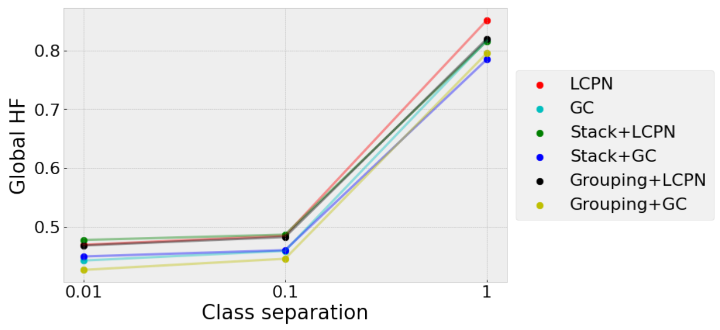


Fig. 7.19: Evolution of the Global HF with respect to the *class separation*. The points of each strategy are the result of averaging the performance on all the scenarios that have the *class separation* parameter equal to the corresponding value on the X axis. The Y axis shows the measure used for comparison (GlobalHf), and the X axis shows the *class separation*.Preferably viewed in color.

of the classifiers to be able to solve an increasing number of classification tasks. Based on the second step analysis, we can only confirm that, based on the *depth of the hierarchies*, we can only aspire to choose between LCPN or GC based strategies, and not between MDHC or HC strategies (Figure 7.22) since it does not make a significant difference.

7.7.6 Effect of *dimensions*

Finally, the first step analysis of the *dimensions* parameter leads us conclude that HC strategies are negatively affected as the number of *dimensions* grows in MDHC problems (Figure 7.23). However, MDHC strategies not only do not become negatively affected, but their overall performance is improved as the *dimensions* grow. This is an expected conclusion since MDHC strategies exploit the dependencies between dimensions, where more dimensions means more possible dependencies which work as hints when classifying an instance. The second step analysis (Figure 7.24) corroborates this effect, every MDHC strategy improves with respect to their HC counterpart, however, it is clearer on the first step analysis.

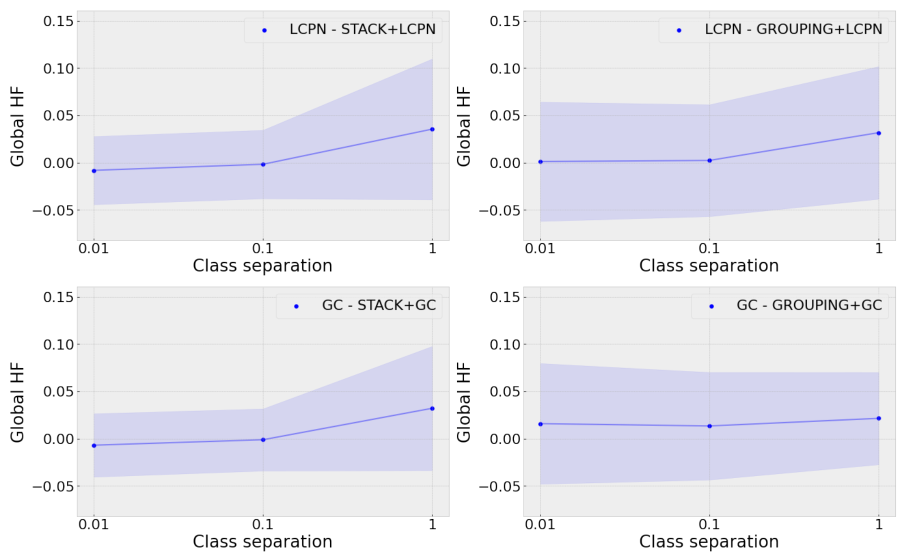


Fig. 7.20: Evolution of the average Global HF difference between the MDHC strategies and their HC counterpart as the *class separation* parameter grows. Each point is calculated as the average of the differences of the performance of a pair of strategies on all the scenarios that have the *class separation* parameter equal to the corresponding value on the X axis. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the *class separation*.

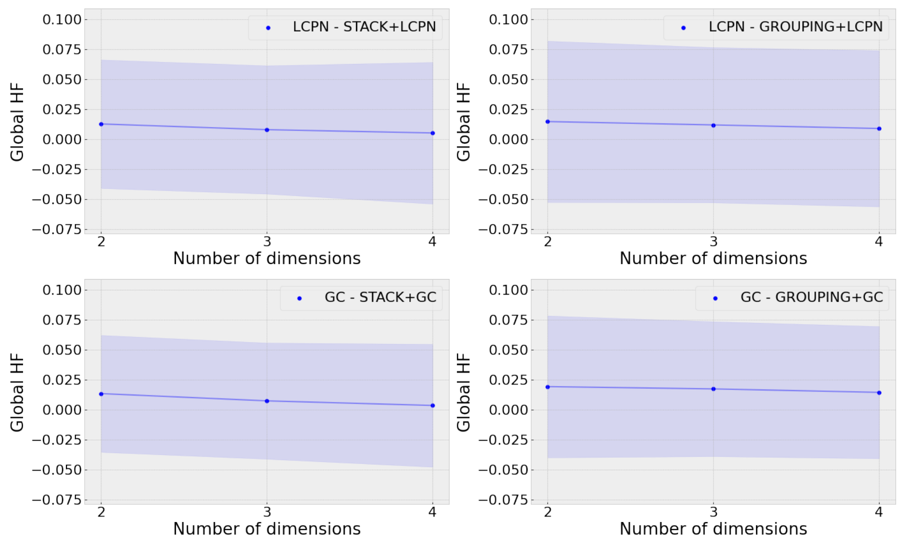


Fig. 7.24: Evolution of the average Global HF difference between the MDHC strategies and their HC counterpart as the *dimensions* parameter grows. Each point is calculated as the average of the differences of the performance of a pair of strategies on all the scenarios that have the *dimensions* parameter equal to the corresponding value on the X axis. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the *dimensions*.

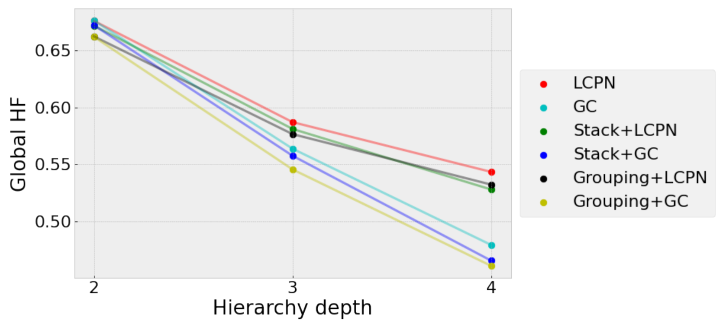


Fig. 7.21: Evolution of the Global HF with respect to the *depth*. The points of each strategy are the result of averaging the performance on all the scenarios that have the *depth* parameter equal to the corresponding value on the X axis. The Y axis shows the measure used for comparison (GlobalHf), and the X axis shows the *depth*. Preferably viewed in color.

The second step of the analysis from Figure 7.20 also shows that the smaller the *class separation* is, the more convenient it is to use MDHC strategies. As seen in Section 7.5, this parameter influences how complicated each classification task is, therefore, the more complicated the classification tasks are, the better it is for MDHC strategies in comparison. However, the improvement of performance as the *class separation* parameter grows takes place mainly between the values 0 and 0.1. This indicates that the relation of this parameter with the overall performance of the classifiers used is not linear.

7.7.7 Confirmation of the two-step univariate analysis

The conclusions derived from the two-step univariate analysis are supported by analyzing some specific scenarios illustrated in Figure 7.25. The performances on the easiest MDHC scenarios (in terms of the scenarios with the parameters where all the strategies perform best: *class separation*=1, *children per parent node*=2, *number of dimensions*=2 and *depth*=2) are illustrated in Figure 7.25a, where we can observe how, for small values of *neurons in the hidden layer* = 2^0, LCPN obtains the best results, since the bottleneck generated severely penalizes the MDHC strategies. However, increasing the number of *neurons in the hidden layer* allows the MDHC strategies to converge to the same level of performance.

Figure 7.25b shows the performances on the most complex scenario with *class separation*=0.01, *children per parent node*=8, *number of dimensions*=4 and *depth*=4. This scenario shows more differences in performance between the strategies, and more interesting behaviours. For small values of *neurons in the hidden layer*, all the strategies perform poorly, however, once the classifiers gain a certain *complexity*, the performance of the Stacking+LCPN strategy

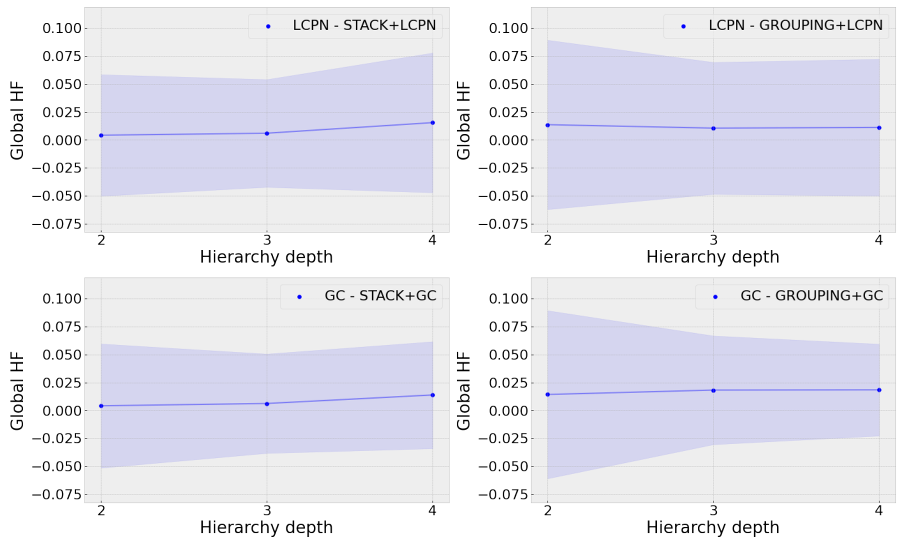


Fig. 7.22: Evolution of the average Global HF difference between the MDHC strategies and their HC counterpart as the *depth* parameter grows. Each point is calculated as the average of the differences of the performance of a pair of strategies on all the scenarios that have the *depth* parameter equal to the corresponding value on the X axis. The Y axes show the measure used for comparison (GlobalHf), and the X axes show the *depth*.

outperforms the other strategies until *neurons in the hidden layer* $= 2^6$ where it is surpassed by Grouping+LCPN.

Having concluded that MDHC strategies perform better on complex MDHC problems in terms of *children per parent node*, *degree of dependency*, *depth*, *number of dimensions* and *class separation*, comparing the MDHC strategies we find out that the LCPN-based strategies perform best overall. However, the downside of using LCPN-based strategies is that they require longer training times, since they need to train more independent models than their GC versions.

7.7.8 Complexity and Training time

Each strategy is composed of a number of classifiers which work as an ensemble, being able to be considered as a single classifier in terms of evaluating their performance. However, in terms of training time, each independent classifier must be trained on a particular task in order to make the whole set work correctly. The total number of classifiers that need to be trained for each strategy (according to the scenarios and definitions of this work) can be calculated as described in Table 7.6:

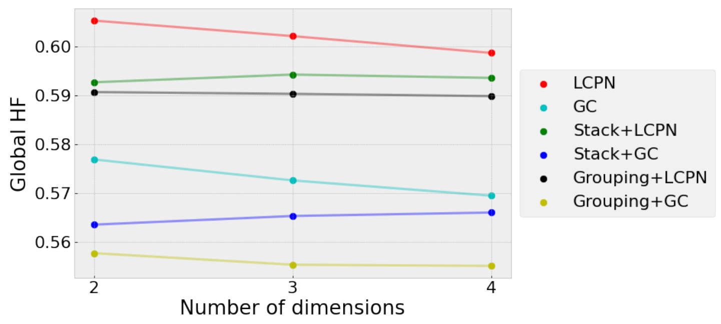
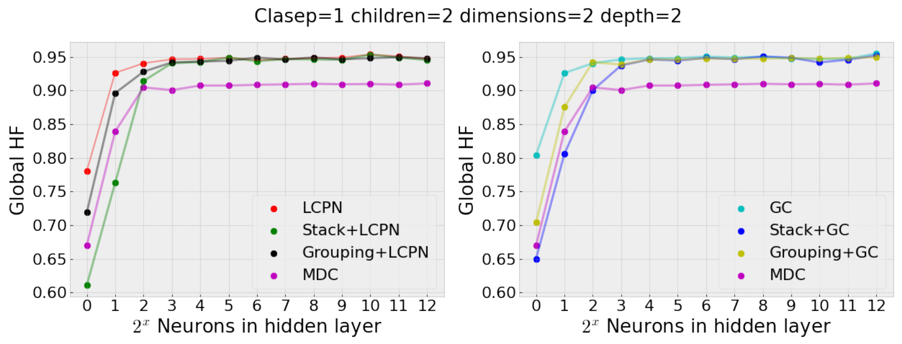


Fig. 7.23: Evolution of the Global HF with respect to the *dimensions*. The points of each strategy are the result of averaging the performance on all the scenarios that have the *dimensions* parameter equal to the corresponding value on the X axis. The Y axis shows the measure used for comparison (GlobalHf), and the X axis shows the *dimensions*. Preferably viewed in color.

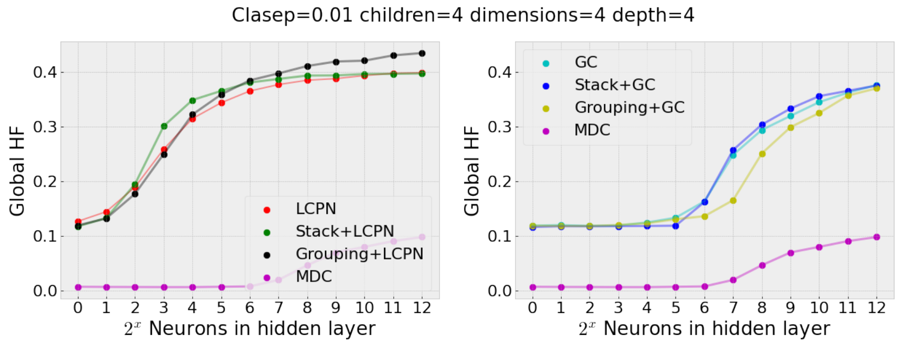
LCPN:	$\#classifiers = dimensions * (1 + children^{depth-1})$
GC:	$\#classifiers = dimensions$
Stacking+LCPN:	$\#classifiers = 2 * dimensions * (1 + children^{depth-1})$
Stacking+GC:	$\#classifiers = 2 * dimensions$
Grouping+LCPN:	$\#classifiers = 1 + children^{depth-1}$
Grouping+GC:	$\#classifiers = 1$
MDC:	$\#classifiers = 1$

Table 7.6: Equations used to calculate the total number of classifiers ($\#classifiers$) that each strategy must train, according to the scenarios and definitions of this work. Where *dimensions* is the number of dimensions of the MDHC problem, *children* is the number of children per parent node of the MDHC problem, and *depth* is the depth of the hierarchies of the MDHC problem.

In Figure 7.26 we can observe how the difference in training time between LCPN-based and GC-based strategies grows for bigger *depth* parameter values. This effect also occurs to a lesser extent with the *children per node* parameter, since both parameters affect the total number of internal nodes, which means more models to be trained.



(a)



(b)

Fig. 7.25: Performance in extreme (in terms of parameter values) scenarios of LCPN-based strategies. The plots in the upper row show the performance of the strategies (LCPN-based on the left, GC-based on the right) on the scenarios with the characteristics listed in the title shared between them, which are considered as "easy" MDHC scenarios. The plots in the bottom row are similar to the previous ones, but on a different scenario considered as "hard" MDHC scenarios. Preferably viewed in color.

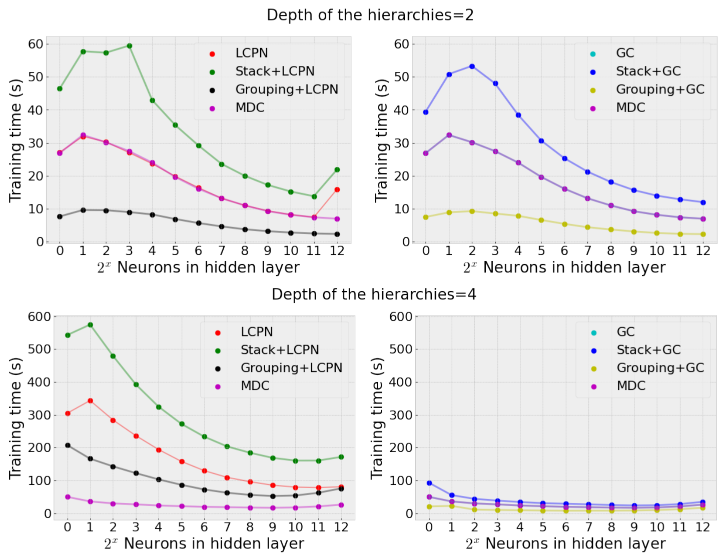


Fig. 7.26: Overall training time of the different strategies on scenarios with different *depth* values. The plots in the upper row show the execution time of the strategies (LCPN-based on the left, GC-based on the right) on the scenarios with the parameter *depth* = 2. The plots in the bottom row on scenarios with the parameter *depth* = 4. The Y axes represent the total execution time in seconds of all the classifiers that are part of each strategy, while the X axes show the *neurons in the hidden layer* parameter. Preferably viewed in color.

Figure 7.26 shows how LCPN-based MDHC strategies require more computation time as the hierarchy size grows. This can be a limiting factor on scenarios with big hierarchies, since the number of internal nodes determines how many models must be trained.

Grouping based methods, on the other hand, do not suffer a priori from this computation time limitation, since there are fewer models to be trained on this strategy, however, they require more complex but fewer models to be trained in order to reach competitive results. This can be a limiting factor when the problem is so complicated that the models inside a Grouping-based strategy need to be too big to be trained on a particular computer.

Taking the training time into account, the selection of the most suitable strategy for a given problem becomes a complex task. Although Stacking+LCPN and Grouping+LCPN present the best results overall, Stacking+LCPN is the strategy that requires the most training time of all, whilst Grouping+LCPN requires a previous study on label dependencies in order to decide what labels must be grouped. Therefore, choosing among these strategies depends on the characteristics of the problem and the characteristics of the training environment.

7.7.9 Empathic datasets results

Regarding the results of the experiments on the Empathic dataset, Table 7.7 shows the average results of a 10-iteration training of the different strategies. The results for the HC strategies are found in the two first columns, highlighted with a white background, while the results for the MDHC strategies are shown in the following four columns, highlighted with a green background.

The results show how every MDHC strategy improves or equals the performance of HC strategies, and that this improvement is achieved for each MDHC performance measure considered. Regarding MDHC strategies, *Stacking* strategies show a better performance than the Grouping strategies, similarly to the results of the experiments on some synthetic scenarios. We could have expected Grouping+LCPN to perform even better than Stacking+LCPN, as seen on many synthetic scenarios, however, the decision making on what labels to group could be decisive in order to get the best performance.

This proves that MDHC strategies exploit the dependencies of MDHC problems, and provide an indirect measure of the *conditional dependencies* of the classification problem [103]. However, the dominance of the MDHC strategies on the Empathic dataset was not expected based on the experiments on synthetic scenarios, where in some cases HC strategies are superior in terms of performance than some MDHC strategies, such as GC and Grouping+GC strategies. A possible explanation for the unexpected MDHC dominance is that the dependencies between hierarchies of a Text-MDHC can be more influential than the simulated dependencies of the synthetic scenarios.

In the case of the Empathic MDHC problem, one aspect that can be contributing to this MDHC dominance is that being the dataset obtained from

	LCPN	**GC**	**Stacking LCPN**	**Grouping LCPN**	**Stacking GC**	**Grouping GC**
$\overline{HP}$	0.60	0.55	**0.63**	0.62	0.62	0.59
$\overline{HR}$	0.61	0.58	**0.65**	0.63	0.64	0.60
$\overline{HF}$	0.57	0.50	**0.63**	0.61	0.62	0.58
$\widehat{HP}$	0.53	0.50	**0.60**	0.57	0.59	0.53
$\widehat{HR}$	0.59	0.58	**0.64**	0.61	0.63	0.60
$\widehat{HF}$	0.56	0.50	**0.61**	0.59	0.60	0.57

Table 7.7: Comparison of MDHC strategies in the Empathic “Topic” and “Intent” MDHC problem. Where $\overline{HP}$, $\overline{HR}$ and $\overline{HF}$ are Mean Hierarchical Precision, Recall and F measures, and $\widehat{HP}$, $\widehat{HR}$ and $\widehat{HF}$ are Global Hierarchical Precision, Recall and F measures introduced in Section 7.3

coaching sessions, centered on some specific topics (Nutrition, Sport and Family), it is expected to have strong dependencies. For example, we expect sentences to have *informative intention* when speaking about *nutrition* or *habits*, whilst if sentences denote *greetings*, they are probably not of those topics. Moreover, each dimension requires a different classifier specialization, that is, for “Topic” classification, the detection of specific words or combination of words is enough to determine the label, whilst “Intent” classification requires a Syntactic Analysis of the sentences [79]. This difference in the specialization of the tasks could prevent a model from partially learning both specializations, resulting in an advantage for the MDHC strategies, which have multiple models with different specializations sharing information between them.

7.8 Conclusions and future work

In this work we have presented and studied the MDHC problem formally. Also, based on previous studies on MDC and HC, we have proposed MDHC performance measures and classification strategies that exploit the dependencies. As a last contribution, in order to evaluate the performance of the strategies presented, we have designed a procedure for creating synthetic MDHC problems with the desired dependencies between labels. Some of the insights from our analysis are the following:

1. MDHC strategies perform better than HC strategies as the complexity of the MDHC classification problem grows in terms of the difficulty of each partial classification task, cardinality of the set of labels of each class variable or dependency between labels.
2. When the model used for the MDHC classification tasks is simple in terms of complexity, HC strategies tend to perform better than MDHC, although in this case making more complex models leads to a better performance regardless of the strategy used.

3. The MDHC LCPN-based strategies outperform the rest of the strategies although they require more training time.

An experiment on a real MDHC problem reinforces the conclusions obtained from the experiments on synthetic scenarios regarding the performance of the different strategies. However, the performance of the MDHC strategies has exceeded our expectations based on the results of the experiments on synthetic datasets, where the improvements where smaller. Although we have made a big effort in defining a clear process with multiple parameters to generate synthetic scenarios with different characteristics, more realistic features could be added in future studies. For example, the features from different dimensions are separable from each other, while in real datasets it is common to have information from different dimensions mixed in the same set of features. This could be achieved by combining features with different operations, but we considered that the study of how these combinations affect the performance is beyond the scope of this work. Also, more complex patterns of dependencies could be developed and studied.

Furthermore, despite having MDHC strategies that perform better on MDHC problems versus traditional HC in many scenarios, we consider that there is room for improvement. A combination of both MDHC strategies could be tested, creating a stacking strategy where the first level of the stack is formed by a Grouping+LCPN strategy, while the second level could be formed by a Stacking+LCPN strategy.

General Conclusions and Future Work

This chapter summarizes the main contributions of this dissertation and outlines potential directions for further research in the fields of NLP and HC.

8.1 Conclusions

The field of NLP has gained widespread recognition owing to the capabilities of language models in addressing various everyday tasks, such as information retrieval and text generation, through written natural language queries. However, several other NLP applications are still in the process of advancing to achieve similar levels of popularity and success.

Virtual assistants have limited presence in our society, primarily restricted to systems that address only a specific range of tasks upon particular requests. The European H2020 EMPATHIC project aimed to research, innovate, explore, and validate new interaction paradigms and platforms for future generations of personalized virtual coaches designed to assist the elderly and their caregivers in achieving active aging goals within the comfort of their homes. This necessitated a system with the capacity not only to comprehend specific queries but also to actively influence user behavior while monitoring and measuring mental health-related parameters and tracking progress toward coaching goals.

In this thesis, we have made significant contributions to various fields related to the EMPATHIC project. Below, we provide a more detailed overview of the specific contributions arising from this dissertation.

In Chapter 3, we introduce an ASR-M simulator capable of simulating transcriptions and errors. This simulator represents a pioneering effort to investigate the impact of various errors generated by ASR-M on the challenging task of End-Of-Turn Detection (EOTD-M). By offering a means to explore and train with simulated errors, this simulator proves instrumental in advancing the field and played a pivotal role in the EOTD task of the Virtual Coach developed as part of the EMPATHIC project.

In Chapter 4, in order to enhance virtual assistants employing coaching strategies, we propose a comprehensive dialog act taxonomy tailored to facilitate communication guided by a coaching strategy. This taxonomy is particularly crucial within the EMPATHIC framework, as it addresses the essential need for implementing a proactive agent that offers assistance and counseling to elderly users, steering dialogs with the objective of achieving coaching goals. This approach distinguishes it from other methods, such as task-oriented dialog systems and chit-chat implementations.

Another notable aspect of the proposed taxonomy is its multimodal nature, with tags organized hierarchically. This characteristic has resulted in contributions to subfields stemming from Hierarchical Classification, specifically Weakly Supervised Hierarchical Classification (WHC) and Multi-Dimensional Hierarchical Classification (MDHC).

In Chapter 6, on our exploration of WHC, we introduce a weakly supervised strategy that incorporates hierarchical information during the training phase. Comparing this strategy to a simpler version without hierarchical information, we find that the inclusion of hierarchical information consistently improves or matches performance across various scenarios. These introduced hierarchical strategies not only enhance classification accuracy but also lead to substantial reductions in computational time, especially in scenarios with extensive label hierarchies.

Finally, in Chapter 7, we formally present and analyze the MDHC problem. Building upon prior research in Multi-Dimensional Classification (MDC) and Hierarchical Classification (HC), we introduce MDHC performance metrics and classification strategies that leverage label dependencies. Our findings underscore the superiority of MDHC strategies in addressing complex MDHC problems marked by challenging extensive label sets, or label dependencies. Notably, MDHC strategies based on Local-Classifier per Parent Node (LCPNs) outperform alternative strategies, albeit with the trade-off of longer training times. Additionally, we introduce a novel procedure for assessing strategy performance by generating synthetic MDHC problems tailored to specific label dependencies, contributing to further advancements in the field.

8.2 Future work

The contributions of this dissertation have led multiple open paths for future research. In the following paragraphs, we present a relation of topics.

Despite the successful utility of the ASR-SIM during the development of the EOTD task, we believe that there are still some aspects that deserve further research and usage:

- **Customized Noise Profiles:** Currently, the ASR-SIM applies noise uniformly across the dataset, assuming a consistent source of noise originating from the shared environment. However, recognizing that individual speakers and recording sessions may introduce variations in noise levels, future

versions of the ASR-SIM should account for these distinct characteristics by tailoring noise profiles to different speech profiles within the dataset.
- **Word Characteristic-Based Errors:** Enhancing the ASR-SIM to generate errors based on specific word characteristics that render certain words more error-prone than others could significantly improve the realism of simulations. Investigating which word characteristics have the most substantial impact on transcription errors can aid in creating more accurate and contextually relevant scenarios.
- **Expansion of Simulation Variables:** Expanding the capabilities of the ASR-SIM beyond simulating pause and word duration variations to include other variables such as tone and audio-derived factors will increase its utility across various algorithms and scenarios. This broader scope will enable researchers to explore a wider range of ASR-related challenges and innovations.
- **Additional Functionalities:** Beyond improving EOTD modules, the ASR-SIM can be leveraged to develop other functionalities within the ASR domain. For example, it can be utilized to address "Confused words", as proposed by Tam et al. [102], who drew inspiration from our ASR-SIM. This highlights the potential for the simulator to catalyze advancements in multiple areas of Automatic Speech Recognition.

Building upon the proposed dialog-act taxonomy, the annotated corpora represents invaluable resources distinguished by their unique characteristics and their origin from an elderly population. The adaptability of our taxonomy extends its relevance beyond coaching-oriented virtual agents, and can be useful for a wide range of conversational scenarios, both general and specialized. The multi-dimensional hierarchical structure of the taxonomy allows for flexible extensions, incorporating additional dimensions to address broader conversations or increased depth to tackle more specific ones. This structural versatility positions it as an ideal foundation for future research endeavors.

Furthermore, our taxonomy proves versatile and applicable to various systems and domains, such as:

- **Computer-Based Job Interview Training [2]:** The taxonomy can be employed in computer-based job interview training systems to enhance dialogue interactions, providing valuable coaching and guidance to job seekers, and monitor interviewed speech.
- **Quality Analysis of Remote Working Experience [77]:** Analyzing interviews with remote workers for the assessment of the quality of their working experience can benefit from the taxonomy's structured approach to dialogue classification, automating the process of evaluating remote work-related conversations.

By extending the reach of the taxonomy to these diverse applications, we unlock its potential to enhance dialogue interactions and automated assessments in various domains, thereby contributing to more effective and efficient communication and analysis.

As task-oriented systems continue to adopt or extend our dialog-act taxonomy, more datasets will be labeled, potentially giving rise to additional weakly-supervised hierarchical classification (WSC) scenarios. The inherent cost and complexity of manually labeling text data often necessitate weakly-supervised approaches. In this evolving landscape, it is likely that our proposed WSC strategy will find adaptation or that new strategies will emerge to address these scenarios. Exploring and comparing different WSC strategies will not only expand the toolkit available for handling hierarchical information but will also enable evaluations in scenarios involving labels that are not leaf nodes in the hierarchy.

In the field of Multi-Dimensional Hierarchical Classification (MDHC), numerous avenues for further research beckon:

- **Feature Integration Across Dimensions:** One promising trajectory involves amalgamating features from different dimensions that are inherently intertwined. This entails incorporating information from diverse dimensions into the same set of features, mirroring the intricate complexities commonly encountered in real-world data.
- **Exploration of Complex Label Dependencies:** Researchers can embark on an exploration of more intricate and nuanced patterns of label dependencies. This entails delving deeper into the interplay of labels and their multifaceted relationships, pushing the boundaries of our understanding in this field.
- **Stacked MDHC Strategies:** Experimentation with a combination of the two MDHC strategies proposed could yield valuable insights into enhanced classification methodologies. Formulating a stacking strategy that integrates a first-level Grouping+LCPN strategy with a second-level Stacking+LCPN strategy has the potential to optimize classification performance and broaden the applicability of MDHC in diverse scenarios.

In order to conclude this section, we propose a more general research line for future work. An exciting avenue for research involves the fusion of Weakly Supervised Classification (WSC) with Multi-Dimensional Hierarchical Classification (MDHC). This fusion has the potential to give rise to entirely novel weakly supervised classification scenarios, characterized by a unique combination of hierarchical structures, weak supervision, and label dependencies. Such an integration promises to introduce fresh challenges and opportunities, paving the way for innovative approaches to address complex classification tasks that bridge multiple dimensions of information and supervision.

8.3 Publications

The research work carried out during this thesis has produced the following publications and submissions:

8.3.1 Referred journals

- **Montenegro, C**., López Zorrilla, A., Mikel Olaso, J., Santana, R., Justo, R., Lozano, J. A., Torres, M. I. (2019). A dialogue-act taxonomy for a virtual coach designed to improve the life of elderly. Multimodal Technologies and Interaction, 3(3), 52.
- **Montenegro, C**., Santana, R., Lozano, J. A. (2021). Analysis of the sensitivity of the End-Of-Turn Detection task to errors generated by the Automatic Speech Recognition process. Engineering Applications of Artificial Intelligence, 100, 104189.
- **Montenegro, C**., Santana, R., Lozano, J. A. (2023). Introducing multidimensional hierarchical classification: Characterization, solving strategies and performance measures. Neurocomputing, 533, 141-160.
- **Montenegro, C**., Santana, R., Lozano, J. A. (2023). Top-down Learning Approach for Weakly Supervised Hierarchical Classification Problems. Submited

8.3.2 Conference communications

- **Montenegro, C**., Santana, R., Lozano, J. A. (2020, July). Transfer learning in hierarchical dialogue topic classification with neural networks. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.
- **Montenegro, C**., Santana, R., Lozano, J. A. (2019, June). Data generation approaches for topic classification in multilingual spoken dialog systems. In Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments (pp. 211-217).

8.3.3 Collaborations

- Torres, M. I., Olaso, J. M., **Montenegro, C**., Santana, R., Vázquez, A., Justo, R., ... Gonzalez-Pinto, A. (2019, June). The empathic project: mid-term achievements. In Proceedings of the 12th ACM International Conference on Pervasive Technologies Related to Assistive Environments (pp. 629-638).
- Gonzalez-Fraile, E., Gonzalez-Pinto, A., Tenorio-Laranga, J., Fernandez-Ruanova, B., Olaso, J. M., **Montenegro, C**., Gordeeva, O. (2020). Empathic, expressive, advanced virtual coach to improve independent healthy-life-years of the elderly (the empathic project: mid-term achievements). European Psychiatry, 63.
- Olaso, J. M., Vázquez, A., Ben Letaifa, L., De Velasco, M., Mtibaa, A., Hmani, M. A., Petrovska-Delacrétaz, D., Chollet, G., **Montenegro, C**., Schlögl, S. (2021, October). The empathic virtual coach: A demo. In Proceedings of the 2021 International Conference on Multimodal Interaction (pp. 848-851).

Appendixes

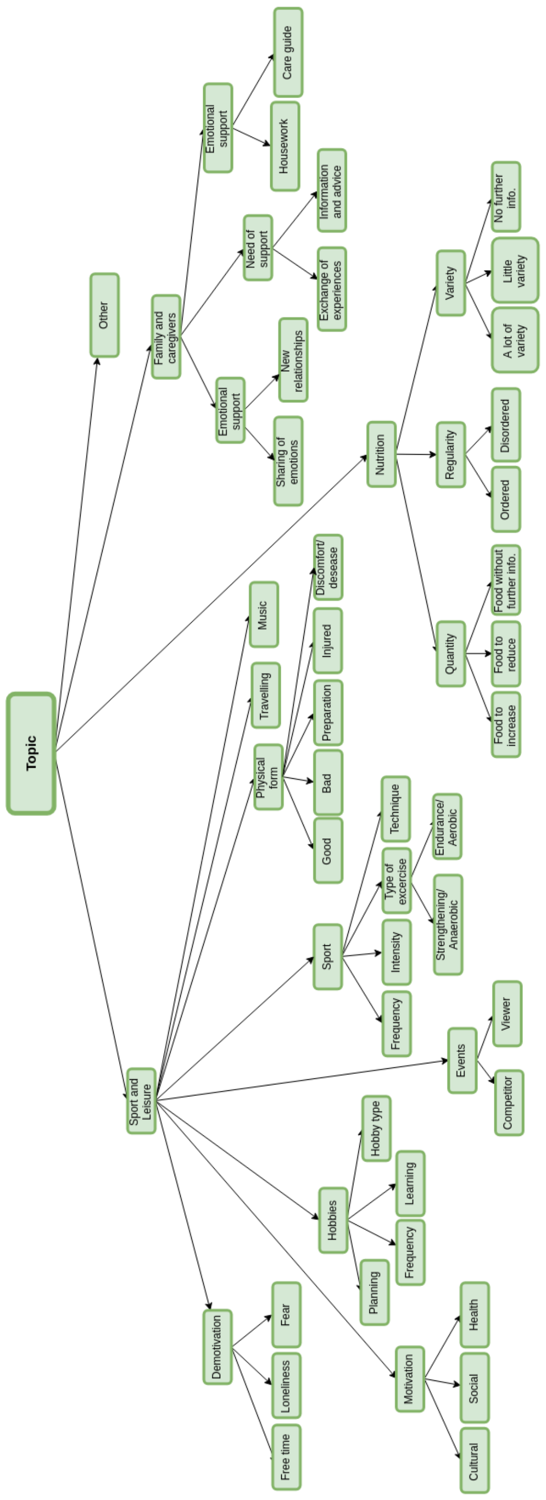


Fig. .1: Topic label tree.

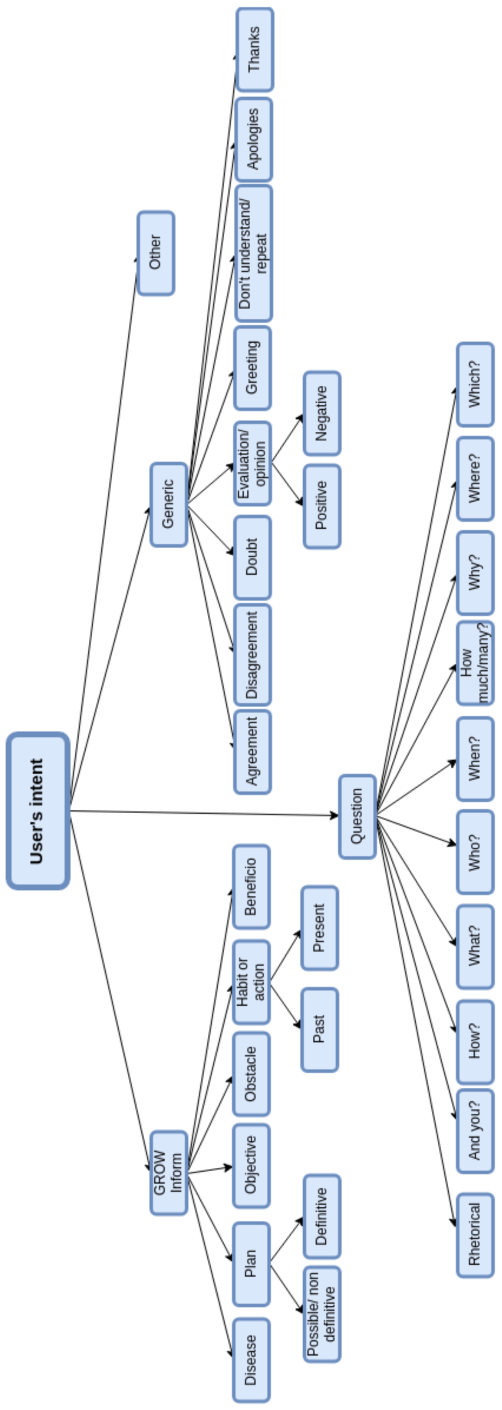


Fig. .2: Intent label tree.

References

[1] Abdulla, W. H., Chow, D., and Sin, G. (2003). Cross-words reference template for DTW-based speech recognition systems. In *TENCON Conference on convergent technologies for Asia-Pacific region*, volume 4, pages 1576–1579. IEEE.

[2] Adiani, D., Colopietro, K., Wade, J., Migovich, M., Vogus, T. J., and Sarkar, N. (2023). Dialogue act classification via transfer learning for automated labeling of interviewee responses in virtual reality job interview training platforms for autistic individuals. *Signals*, 4(2):359–380.

[3] Aldeneh, Z., Dimitriadis, D., and Provost, E. M. (2018). Improving end-of-turn detection in spoken dialogues by detecting speaker intentions as a secondary task. In *International Conference on Acoustics, Speech and Signal Processing*, pages 6159–6163. IEEE.

[4] Allen, J. and Core, M. (1997). Draft of DAMSL: Dialog act markup in several layers.

[5] Anderson, A. H., Bader, M., Bard, E. G., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., et al. (1991). The HCRC map task corpus. *Language and speech*, 34(4):351–366.

[6] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

[7] Austin, J. L. (1975). *How to do things with words*. Oxford university press.

[8] Bielza, C., Li, G., and Larrañaga, P. (2011). Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727.

[9] Bohus, D. and Rudnicky, A. I. (2009). The ravenclaw dialog management framework: Architecture and systems. *Computer Speech Language*, 23(3):332 – 361.

[10] Bunt, H. (2009a). The dit++ taxonomy for functional dialogue markup. In Decker, Sichman, S. and Castelfranchi, editors, *Proc. of 8th Int. Conf.*

on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary.

[11] Bunt, H. (2009b). The DIT++ taxonomy for functional dialogue markup. In *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pages 13–24.

[12] Bunt, H., Alexandersson, J., Choe, J.-W., Fang, A. C., Hasida, K., Petukhova, V., Popescu-Belis, A., and Traum, D. R. (2012). ISO 24617-2: A semantically-based standard for dialogue annotation. In *LREC*, pages 430–437.

[13] Bunt, H., Petukhova, V., Malchanau, A., Wijnhoven, K., and Fang, A. (2016). The dialogbank. In Chair), N. C. C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

[14] Bunt, H., Petukhova, V., Traum, D., and Alexandersson, J. (2017). *Dialogue Act Annotation with the ISO 24617-2 Standard*, pages 109–135. Springer International Publishing, Cham.

[15] Burred, J. J. and Lerch, A. (2003). A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th International Conference on Digital Audio Effects*, pages 8–11.

[16] Campione, E. and Véronis, J. (2002). A large-scale multilingual study of silent pause duration. In *Speech Prosody, International Speech and Communication Association*, pages 199–202.

[17] Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P. (1998). Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB journal*, 7(3):163–178.

[18] Chang, S.-Y., Li, B., Sainath, T. N., Simko, G., and Parada, C. (2017). Endpoint detection using grid long short-term memory networks for streaming speech recognition. pages 3812–3816.

[19] Cour, T., Sapp, B., and Taskar, B. (2011). Learning from partial labels. *The Journal of Machine Learning Research*, 12:1501–1536.

[20] Dembczynski, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2010). On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-label Data*, pages 5–12.

[21] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: series B (methodological)*, 39(1):1–22.

[22] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[23] Elidan, G. (2011). Bagged structure learning of bayesian network. In *Proceedings of the Fourteenth International Conference on Artificial In-*

telligence and Statistics, pages 251–259. JMLR Workshop and Conference Proceedings.

[24] Fossati, D. and Di Eugenio, B. (2008). I saw tree trees in the park: How to correct real-word spelling mistakes. In *LREC*, pages 896–901.

[25] Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992a). Switchboard: Telephone speech corpus for research and development. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 517–520. IEEE.

[26] Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992b). Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

[27] Graham, A. (2006). Behavioural coaching - the GROW model. In *Passmore, Jonathan. Excellence in coaching: the industry guide (2nd ed.)*, pages 83–93.

[28] Grant, A. M. (2003). The impact of life coaching on goal attainment, metacognition and mental health. *Social Behavior and Personality*, 31(3):253–263.

[29] Grant, A. M. (2011). Is it time to regrow the grow model? issues related to teaching coaching session structures. *The Coaching Psychologist*, 7(2):118–126.

[30] Gravano, A. and Hirschberg, J. (2011). Turn-taking cues in task-oriented dialogue. *Computer Speech & Language*, 25(3):601–634.

[31] Gupta, P., Banchs, R. E., and Rosso, P. (2016). Squeezing bottlenecks: exploring the limits of autoencoder semantic representation capabilities. *Neurocomputing*, 175:1001–1008.

[Guyon] Guyon, I. Design of experiments for the nips 2003 variable selection benchmark 2003. *NIPS2003*.

[33] Hakkani-Tür, D., Tür, G., Çelikyilmaz, A., Chen, Y., Gao, J., Deng, L., and Wang, Y. (2016). Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In Morgan, N., editor, *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 715–719. ISCA.

[34] Hara, K., Inoue, K., Takanashi, K., and Kawahara, T. (2019). Turn-taking prediction based on detection of transition relevance place. pages 4170–4174.

[35] Heck, L. and Hakkani-Tür, D. (2012). Exploiting the semantic web for unsupervised spoken language understanding. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 228–233.

[36] Hernández, J., Sucar, L. E., and Morales, E. F. (2014). Multidimensional hierarchical classification. *Expert systems with applications*, 41(17):7671–7677.

[37] Hernández-González, J., Inza, I., and Lozano, J. A. (2013). Learning bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440.

[38] Hernandez-Leal, P., Orihuela-Espina, F., Sucar, E., and Morales, E. F. (2012). Hybrid binary-chain multi-label classifiers. In *Proceedings of the 6th European Workshop Probabilistic Graphical Models*, pages 139–146.

[39] Jones, R., Woods, S., and Guillaume, Y. (2015). The effectiveness of workplace coaching: A meta-analysis of learning and performance outcomes from coaching. *Journal of Occupational and Organizational Psychology*, 89:249.277.

[40] Justo, R., Ben Letaifa, L., Palmero, C., Gonzalez-Fraile, E., Torp Johansen, A., Vázquez, A., Cordasco, G., Schlögl, S., Fernández-Ruanova, B., Silva, M., et al. (2020). Analysis of the interaction between elderly people and a simulated virtual coach. *Journal of Ambient Intelligence and Humanized Computing*, 11:6125–6140.

[41] Keizer, S., Bunt, H., and Petukhova, V. (2011). Multidimensional dialogue management. In *Interactive Multi-modal Question-Answering*, pages 57–86. Springer.

[42] Kiritchenko, S., Matwin, S., Famili, A. F., et al. (2005). Functional annotation of genes using hierarchical text categorization. In *In Proceedings of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, pages 1–4.

[43] Koller, D. and Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178.

[44] Kumar, E. (2013). *Natural language processing*. IK International Pvt Ltd.

[45] Kumar, S. and Rowley, H. A. (2007). Classification of weakly-labeled data with partial equivalence relations. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

[46] Lampert, A., Dale, R., and Paris, C. (2006). Classifying speech acts using verbal response modes. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 34–41.

[47] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

[48] Lewis, D. D., Yang, Y., Russell-Rose, T., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397.

[49] Li, T. and Ogihara, M. (2005). Music genre classification with taxonomy. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 5, pages v–197. IEEE.

[50] Liu, C., Ishi, C., and Ishiguro, H. (2017). Turn-taking estimation model based on joint embedding of lexical and prosodic contents. In *Proceedings of Interspeech*, pages 1686–1690.

[51] Liu, X., Zhou, Y., and Zheng, R. (2007). Sentence similarity based on dynamic time warping. In *International Conference on Semantic Computing*, pages 250–256. IEEE.

[52] López Zorrilla, A., Velasco Vázquez, M. d., Irastorza, J., Olaso Fernández, J. M., Justo Blanco, R., and Torres Barañano, M. I. (2018). EMPATHIC: Empathic, expressive, advanced virtual coach to improve independent healthy-life-years of the elderly. *Procesamiento del Lenguaje Natural*, 61:167–170.

[53] Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *CoRR*, abs/1506.08909.

[54] Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., and Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313.

[55] Maier, A., Hough, J., and Schlangen, D. (2017). Towards deep end-of-turn prediction for situated spoken dialogue systems. pages 1676–1680.

[56] Masumura, R., Asami, T., Masataki, H., Ishii, R., and Higashinaka, R. (2017). Online end-of-turn detection from speech based on stacked time-asynchronous sequential networks. In *Proceedings of Interspeech*, pages 1661–1665.

[57] Masumura, R., Tanaka, T., Ando, A., Ishii, R., Higashinaka, R., and Aono, Y. (2018). Neural dialogue context online end-of-turn detection. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 224–228.

[58] McKay, C. and Fujinaga, I. (2004). Automatic genre classification using large high-level musical feature sets. In *ISMIR*, volume 2004, pages 525–530.

[59] Meng, Y., Shen, J., Zhang, C., and Han, J. (2019). Weakly-supervised hierarchical text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6826–6833.

[60] Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

[61] Montenegro, C., López Zorrilla, A., Mikel Olaso, J., Santana, R., Justo, R., Lozano, J. A., and Torres, M. I. (2019a). A dialogue-act taxonomy for a virtual coach designed to improve the life of elderly. *Multimodal Technologies and Interaction*, 3(3):1–52.

[62] Montenegro, C., Santana, R., and Lozano, J. (2023). Introducing multi-dimensional hierarchical classification: Characterization, solving strategies and performance measures. *Neurocomputing*, 533:141–160.

[63] Montenegro, C., Santana, R., and Lozano, J. A. (2019b). Data generation approaches for topic classification in multilingual spoken dialog systems. In *12th Conference on PErvasive Technologies Related to Assistive Environments Conference (PETRA-19)*. ACM.

[64] Morris, A. C., Maier, V., and Green, P. (2004). From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*, pages 2765–2768.

[65] Naik, A., Charuvaka, A., and Rangwala, H. (2013). Classifying documents within multiple hierarchical datasets using multi-task learning. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 390–397. IEEE.

[66] Narayanan, S., Georgiou, P. G., Sethy, A., Wang, D., Bulut, M., Sundaram, S., Ettelaie, E., Ananthakrishnan, S., Franco, H., and Precoda, K. e. a. (2006). Speech recognition engineering issues in speech to speech translation system design for low resource languages and domains. In *International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 1209–1212. IEEE.

[67] of Europe Development Bank (CEB), C. (2014). Ageing populations in europe: Challenges and opportunities for the ceb. *JAMA*, pages 1–82.

[68] Pareti, S. and Lando, T. (2018). Dialog intent structure: A hierarchical schema of linked dialog acts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).*

[69] Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.-R., and Galinari, P. (2015). Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, pages 1–9.

[70] Passmore, J. (2011). Motivational interviewing – a model for coaching psychology practice. *The Coaching Psychologist*, 7(1):35–39.

[71] Passmore, J. (2012). An integrated model of goal-focused coaching: an evidence-based framework for teaching and practice. *International Coaching Psychology Review*, 7(2):146–165.

[72] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., and Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072.

[73] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[74] Perello-Nieto, M., Santos-Rodriguez, R., Garcia-Garcia, D., and Cid-Sueiro, J. (2020). Recycling weak labels for multiclass classification. *Neurocomputing*, 400:206–215.

[75] Petukhova, V. and Bunt, H. (2012). The coding and annotation of multimodal dialogue acts. In *LREC*, pages 430–437.

[76] Popescu-Belis, A. (2005). Dialogue acts: One or more dimensions. *ISSCO WorkingPaper*, 62.

[77] Porcu, S., Floris, A., and Atzori, L. (2022). Analysis of the quality of remote working experience: a speech-based approach. *Quality and User Experience*, 7(1):2.

[78] Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A. K., and Webber, B. L. (2008). The penn discourse treebank 2.0. In *LREC*, pages 1–8.

[79] Purohit, H., Dong, G., Shalin, V., Thirunarayan, K., and Sheth, A. (2015). Intent classification of short-text on social media. In *2015 IEEE international conference on smart city/socialcom/sustaincom (smartcity)*, pages 222–228. IEEE.

[80] Qadir, A. and Riloff, E. (2011). Classifying sentences as speech acts in message board posts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 748–758. Association for Computational Linguistics.

[81] Razavi, S. Z., Kane, B., and Schubert, L. K. (2019). Investigating linguistic and semantic features for turn-taking prediction in open-domain human-computer conversation. pages 4140–4144.

[82] Read, J., Bielza, C., and Larrañaga, P. (2013). Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1720–1733.

[83] Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*.

[84] Roddy, M., Skantze, G., and Harte, N. (2018). Investigating speech features for continuous turn-taking prediction using LSTMs. *arXiv preprint arXiv:1806.11461*.

[85] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243. IEEE.

[86] Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., et al. (2004). The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545.

[87] Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

[88] Sandhaus, E. (2008). The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

[89] Santos, A. and Canuto, A. (2014). Applying semi-supervised learning in hierarchical multi-label classification. *Expert Systems with Applications*, 41(14):6075–6085.

[90] Sayas, S. (2018a). Dialogues on leisure and free time. Technical Report DP3, Empathic project.

[91] Sayas, S. (2018b). Dialogues on nutrition. Technical Report DP1, Empathic project.

[92] Sayas, S. (2018c). Dialogues on physical exercise. Technical Report DP2, Empathic project.

[93] Serban, I. V., Lowe, R., Henderson, P., Charlin, L., and Pineau, J. (2015). A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742.

[94] Serdyuk, D., Audhkhasi, K., Brakel, P., Ramabhadran, B., Thomas, S., and Bengio, Y. (2016). Invariant representations for noisy speech recognition. *arXiv preprint arXiv:1612.01928*.

[95] Serrano-Pérez, J. and Sucar, L. E. (2021). Artificial datasets for hierarchical classification. *Expert Systems with Applications*, 182:115218.

[96] Shannon, M., Simko, G., Yiin Chang, S., and Parada, C. (2017). Improved end-of-query detection for streaming speech recognition. In *Proceedings of Interspeech*, pages 1909–1913.

[97] Silla, C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

[98] Simonnet, E., Ghannay, S., Camelin, N., and Estève, Y. (2018). Simulating ASR errors for training SLU systems. In *LREC*, pages 3157–3162.

[99] Singh, M. (1997). Learning bayesian networks from incomplete data. *AAAI/IAAI*, 1001:534–539.

[100] Skantze, G. (2017). Towards a general, continuous model of turn-taking in spoken dialogue using lstm recurrent neural networks. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 220–230.

[101] Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Ess-Dykema, C. V., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

[102] Tam, Y.-C., Xu, J., Zou, J., Wang, Z., Liao, T., and Yuan, S. (2022). Robust unstructured knowledge access in conversational dialogue with asr errors. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6702–6706. IEEE.

[103] Tenenboim-Chekina, L., Rokach, L., and Shapira, B. (2010). Identification of label dependencies for multi-label classification. In *Working Notes of the Second International Workshop on Learning from Multi-Label Data*, pages 53–60.

[104] Theeboom, T., Beersma, B., and van Vianen, A. E. (2014). Does coaching work? a meta-analysis on the effects of coaching on individual level outcomes in an organizational context. *The Journal of Positive Psychology*, 9(1):1–18.

[105] Tomás, J. T., Spolaôr, N., Cherman, E. A., and Monard, M. C. (2014). A framework to generate synthetic multi-label datasets. *Electronic Notes in Theoretical Computer Science*, 302:155–176.

[106] Torres, M. I., Olaso, J. M., Glackin, N., Justo, R., and Chollet, G. (2018). A spoken dialogue system for the empathic virtual coach. In *International Workshop on Spoken Dialog System Technology (IWSDS)*.

[107] Torres, M. L., Olaso, J. M., Montenegro, C., Santana, R., Vazquez, A., Justo, R., Lozano, J. A., Schloegl, S., Chollet, G., Dugan, N., Irvine, M., Glackin, N., Pickard, C., Esposito, A., Cordasco, G., Troncone, A., Petrovska-Delacretaz, D., Mtibaa, A., Hmani, M. A., Korsnes, M. S., Mar-

tinussen, L. J., Escalera, S., Palmero-Cantarino, C., Deroo, O., Gordeeva, O., Tenerio-Laranga, J., Gonzalez-Fraile, E., Fernandez-Ruanova, B., and Gonzalez-Pinto, A. (2019a). The EMPATHIC Project: Mid-term Achievements. In *12th Conference on PErvasive Technologies Related to Assistive Environments Conference (PETRA-19)*. ACM.

[108] Torres, M. L., Olaso, J. M., Montenegro, C., Santana, R., Vazquez, A., Justo, R., Lozano, J. A., Schloegl, S., Chollet, G., Dugan, N., Irvine, M., Glackin, N., Pickard, C., Esposito, A., Cordasco, G., Troncone, A., Petrovska-Delacretaz, D., Mtibaa, A., Hmani, M. A., Korsnes, M. S., Martinussen, L. J., Escalera, S., Palmero-Cantarino, C., Deroo, O., Gordeeva, O., Tenerio-Laranga, J., Gonzalez-Fraile, E., Fernandez-Ruanova, B., and Gonzalez-Pinto, A. (2019b). The EMPATHIC Project: Mid-term Achievements. In *Proceedings of the 12th Conference on PErvasive Technologies Related to Assistive Environments Conference (PETRA-19)*, pages 629–638. ACM.

[109] Traum, D. R. (2000). 20 questions on dialogue act taxonomies. *Journal of semantics*, 17(1):7–30.

[110] Tur, G., Celikyilmaz, A., He, X., Hakkani-Tür, D., and Deng, L. (2018). *Deep Learning in Conversational Language Understanding*, pages 23–48. Springer Singapore, Singapore.

[111] Tur, G. and DeMori, R. (2011). *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech.* John Wiley and Sons.

[112] Vogrinčič, S. and Bosnić, Z. (2011). Ontology-based multi-label classification of economic articles. *Computer Science and Information Systems*, 8(1):101–119.

[113] Voleti, R., Liss, J. M., and Berisha, V. (2019). Investigating the effects of word substitution errors on sentence embeddings. In *International Conference on Acoustics, Speech and Signal Processing*, pages 7315–7319. IEEE.

[114] Vukotic, V., Pintea, S., Raymond, C., Gravier, G., and van Gemert, J. C. (2017). One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network. *CoRR*, abs/1702.04125.

[115] Wang, S., Wang, J., Wang, Z., and Ji, Q. (2014). Enhancing multi-label classification by modeling dependencies among labels. *Pattern Recognition*, 47(10):3405–3413.

[116] Wang, Y. and Acero, A. (2006). Discriminative models for spoken language understanding. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*. ISCA.

[117] Webb, E. C. et al. (1992). *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* Number Ed. 6. Academic Press.

[118] Weilhammer, K. and Rabold, S. (2003). Durational aspects in turn taking. In *Proceedings of the International Conference of Phonetic Sciences*, pages 2145–2148.

[119] Whitemore, J. (2009). *Coaching for performance : growing human potential and purpose : the principles and practice of coaching and leadership.* Nicholas Brealey Publishing, London.

[120] Willcox, D. C., Scapagnini, G., and Willcox, B. J. (2014). Healthy aging diets other than the mediterranean: a focus on the okinawan diet. *Mechanisms of Ageing and Development*, 136:148–162.

[121] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

[122] Wong, M. L. and Guo, Y. Y. (2008). Learning bayesian networks from incomplete databases using a novel evolutionary algorithm. *Decision Support Systems*, 45(2):368–383.

[123] Xiao, H., Liu, X., and Song, Y. (2019). Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification. In *The World Wide Web Conference*, pages 3370–3376.

[124] Yaman, S., Deng, L., Yu, D., Wang, Y., and Acero, A. (2008). An integrative and discriminative technique for spoken utterance classification. *IEEE Trans. Audio, Speech & Language Processing*, 16(6):1207–1214.

[125] Yu, D. and Deng, L. (2016). *Automatic Speech Recognition.* Springer Publishing Company, Incorporated.

[126] Zechner, K. and Waibel, A. (2000). Minimizing word error rate in textual summaries of spoken language. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics, Seatle*, pages 186–193.

[127] Zhang, M.-L. and Zhang, K. (2010). Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 999–1008.

[128] Zhang, R., Li, W., Gao, D., and Ouyang, Y. (2013). Automatic twitter topic summarization with speech acts. *IEEE transactions on audio, speech, and language processing*, 21(3):649–658.

[129] Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., and Weston, J. (2018). Personalizing dialogue agents: I have a dog, do you have pets too? *CoRR*, abs/1801.07243.