University of Saarland
Faculty of Mathematics and Computer Science
Department of Computer Science

Master Thesis

Automatic Assignment of Semantic Frames in Dialogue

submitted by

Natalia Skachkova

Supervisor: **Prof. Dr. Josef van Genabith**
Co-Supervisor: **Ing. Ivana Kruijff-Korbayová, Ph.D.**

January 24, 2021

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis .

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 24. 01. 2021 

(Datum/Date) (Unterschrift/Signature)

Abstract

Automatic Assignment of Semantic Frames in Dialogue

Natural language understanding (NLU) is one of the most challenging NLP tasks. There exist different approaches to 'teaching' a machine to 'understand' the meaning of words, sentences and larger units of text. One of them employs the ideas of frame semantics.

According to frame semantics, a semantic frame represents some event or situation. The meaning of a word can be understood only in context, and both the word and the expressions that introduce the context take certain slots of a frame.

Currently, the application of frame semantics to NLU tasks is not a very well-explored research topic. The task of automatic assignment of semantic frames in dialogue is hardly studied at all. An additional challenge here is a lack of annotated data. However, classification of utterances using frames offers a lot of advantages: it allows to capture the semantics of the whole utterance, gives it structure and has necessary inventory to formalize this structure together with the relations between its elements.

In this thesis we investigate the potential of frame semantics as a meaning representation framework for team communication in a disaster response scenario. We focus on the automatic frame assignment and retrain PAFIBERT, which is one of the state-of-the-art frame classifiers, on English and German TRADR data. We examine the performance of both models and discuss their adjustments, such as sampling of additional instances from an unrelated domain, adding extra features to input token representations, applying frame filtering to exclude unlikely candidate frames and some others.

We show that sampling extra out-of-domain training data has a limited positive effect if the original training set is small, and that filtering may also produce a positive influence on the classifier performance, if the training set is large enough and has a good coverage of frame-evoking targets. We discuss an unexpected impact of extra features on the models' behaviour and perform a careful study of the mistakes made by the classifiers.

In addition, we present a detailed analysis of all the corpora used in our experiments with respect to the distributions of semantic frames, lexical units that evoke them, parts of speech of these units and so on. The results of this analysis can be useful for further research in the area of frame semantics. Finally, we summarize our experience of annotating TRADR data with frames in the form of annotation guidelines.

Contents

Acknowledgements

I wish to express my sincere appreciation to my supervisors, Ing. Ivana Kruijff-Korbayová, Ph.D. and Prof. Dr. Josef van Genabith, who offered me an opportunity to work on this interesting topic, helped me with advice, gave feedback and insightful comments. I would also like to thank my fellow students/colleagues Tatiana Anikina for sharing useful ideas for the experiments and Daria Fedorova for her help with the annotation of the TRADR data and systematization of the correspondences between FrameNet and SALSA frame labels.

Chapter 1

Introduction

Undoubtedly, nowadays natural language processing (NLP) is living through exciting times, characterized by a rapid development and success in various fields, such as syntax, semantics, discourse or speech. Recent advances in neural networks as well as ever-growing computational and memory capacities allowed to train language models that show great results in such tasks as segmentation, parsing, tagging, different sorts of labelling and many others.

There exist models that are good at solving more complex problems, e.g., machine translation, summarization, co-reference resolution, information extraction, content analysis or question answering. Such tasks are closely related to semantics and their full realization assumes processing input in context. Often they imply performing inference.

Now it is also possible to build language models for writing poetry, fiction, programming code and so on. There are models that allow us to talk with some application - to order food, buy tickets, choose a restaurant, a cinema or find the nearest parking place. All this sometimes makes us think that a computer is able to understand spoken or written speech and think like humans do.

How do humans think? And what does it mean - 'to understand'? The answers to these questions lie out of the scope of the present thesis, because the phenomena under examination are rather complex processes. And still, some of their aspects can be analyzed from the point of view of computational linguistics and computer science. They also can be simulated using state of the art machine learning techniques. Such simulation often implies using some meaning representation framework as a means of 'understanding' the meaning of a sentence or an utterance.

In the present work we would like to refer to the theory of frame semantics as a meaning representation framework for dialogues from the domain of disaster response. Our work is actually a part of a larger research project that aims at extracting run-time mission knowledge from verbal team communication in a disaster response scenario. Mission knowledge encompasses the mission goals, tasks that mission participants asked to perform and their status, the relevant objects and locations and so on. Extracting this information and keeping track of changes in a dynamic mission environment can help provide situation awareness and teamwork assistance [Willms et al., 2019].

Frame semantics is a paradigm defining the meaning of words through the context they are used in [Fillmore, 1976]. This assumes that, depending on context, a word (or

an expression) is able to evoke in our minds a certain event or situation together with a set of slots called frame elements associated with it, even if some of these slots were not explicitly filled in the sentence.

Using frame semantics as a full meaning representation structure implies performing frame semantic parsing, namely identifying frame-evoking elements (targets) and corresponding frames, as well as recognizing certain spans as frame elements and classifying them. These are challenging tasks, with the main problems being a lack of training data, a great variety of frames and polysemy of words. In our thesis we will focus only on automatic assignment of semantic frames in English and German TRADR dialogues [Kruijff-Korbayová et al., 2015] given a target. And before doing this, we will also need to study the main characteristics of team communication in our data, and to annotate the corpus with frames and targets that evoke them. So, in this thesis we will try to answer the following research questions:

1. Can we apply existing guidelines for text annotation to dialogue data? Relying on FrameNet annotation guidelines by Ruppenhofer et al. [2006], we will annotate all TRADR dialogues with semantic frames, targets (frame related elements) and lexical units. We will summarize our experience, highlight the differences in the annotation of text and dialogue data, and illustrate the main characteristics of our annotation approach, as well as challenging cases that require further consideration.

2. Is the TRADR corpus much different from other available data annotated with frames, namely FrameNet [Baker and Sato, 2003] and SALSA [Burchardt et al., 2006]? We will present a detailed analysis of these datasets, including their domains and distributions of semantic frames, lexical units of frame-evoking targets and their parts of speech. This will allow us to estimate the possibility of using out-of-domain data for training a frame classifier for team communication in a disaster response scenario.

3. Can the frame distribution influence the performance of a classifier? We will investigate why multiclass imbalanced data requires specific learning approaches and/or specific performance metrics and will give an overview of them.

4. Given a target expression, how to train a classifier to assign frames? Can some existing models be reused or fine-tuned on TRADR data? We will consider several approaches to building a frame classifier for dialogue data from the domain of disaster response. We will start with a simple sequence classification approach that assumes fine-tuning of a pretrained $\text{BERT}_{\text{BASE}}$ model [Devlin et al., 2019] on TRADR data. Next, we will consider reusing one of the existing state-of-the-art frame classifiers trained on the FrameNet text data for English TRADR dialogues, as well as retraining this classifier on our English or German data.

5. How well will a classifier trained on text perform on dialogue data? Will the performance of the classifier change if we train it on mixed data (FrameNet plus TRADR)? We will suggest three different sampling approaches, discuss the mistakes that the classifiers make, and try to explain why they make them. In

contrast to many papers that report standard accuracy, precision, recall and F-score to measure the performance of a frame classifier, we will concentrate on metrics specifically designed for imbalanced data, e.g., geometric mean and the index of balanced accuracy.

6. Is it possible to achieve a better performance of the state-of-the-art classifier trained on TRADR data? If yes, then how? And if no, then why? We will examine several opportunities for the improvement of the classifier performance, such as changing the size of the context window of the frame-evoking targets, enriching the input tokens with lexical and discourse features and applying the frame filtering mechanism.

7. Are there any differences in training a frame classifier on English and German TRADR data? Do sampling and adding extra features have a similar effect on the two classifiers? We will compare the performance of English and German models and try to explain possible differences or similarities.

The structure of the thesis is as follows. In Chapter 2 we will present background information in order to provide the necessary context for our research, such as a survey of meaning representation frameworks, the key points of the frame semantics theory and its realization as the FrameNet project that resulted in the creation of a frame database and the FrameNet corpus. We will also give an overview of other available corpora, both text and dialogue, annotated with frames, and try to summarize the main achievements in frame-semantic parsing at the levels of a separate sentence/utterance and discourse. Next, we will discuss multiclass classification with a focus on neural networks and BERT, as $\text{BERT}_{\text{BASE}}$ makes up the core of all the frame classifiers that we are going to train. Because the datasets we will be working with are not only multiclass but also heavily imbalanced, we will give an overview of the approaches to deal with imbalanced data, and introduce suitable performance measures.

Chapter 3 will be devoted to the analysis of English and German TRADR, as well as FrameNet and SALSA data. We will examine and compare their domains, size, the distributions of semantic frames, lexical units and so on. Additionally, we will explain how we preprocessed all data for the experiments.

In the next chapter we will present the frame classifiers, namely our baseline which is a simple sequence classification model trained on TRADR data, and the PAFIBERT model [Tan and Na, 2019] that we want to reuse. We will investigate how well the PAFIBERT model performs on the TRADR corpus when trained on in-domain and out-of-domain data, analyse the classifier mistakes, introduce several modifications of the basic PAFIBERT model, and discuss where and why they fail or succeed. Where possible, we will run the same experiments in parallel for English and German TRADR data and compare the results.

Finally, in Chapter 5 we will make a conclusion. In addition to that, in Appendix A the interested reader can find the annotation guidelines describing our approach to the annotation of TRADR data with semantic frames, targets, lexical units and so on. Appendix B contains the definitions of new frames that we introduced, as well as examples. Appendix C provides additional information about multiclass classification techniques.

Chapter 2

Background and Related Work

The aim of this chapter is to give the reader the necessary background knowledge about various topics related to automatic assignment of semantic frames in dialogue. We will start with a definition of semantic parsing and a brief overview of meaning representation frameworks. This will help familiarize the reader with various approaches to capturing the meaning of a sentence or an utterance, and illustrate the place of frame semantics among them. We will explain how frame semantics is different from these approaches, and why we decided to choose it as a meaning representation framework. The next section will present the FrameNet project as a practical realization of frame semantics, and provide the reader with necessary terminology and examples. In addition, we will also give an overview of various corpora annotated with semantic frames and introduce a number of frameworks designed to perform frame-semantic parsing. As the goal of our work is to train a frame classifier attempting to differentiate between hundreds of labels, we will also discuss neural networks as one of the most common approaches to multiclass classification. We will focus on the BERT model, which will be used as a core element of all our models that will be presented later. Finally, we will summarize the most common approaches to working with imbalanced data, explain why some typical performance metrics should be avoided when working with it, and introduce the metrics that we are going to use instead.

2.1 Meaning representation frameworks

Before we proceed with the topic of frame semantics, we would like to explain in more detail what semantic parsing in general is, what approaches to the meaning representation of a sentence/utterance or even a larger piece of discourse exist, and also to compare frame semantics with these approaches.

Kate and Wong [2010] define semantic parsing as transforming natural language sentences into computer executable complete meaning representations for domain-specific applications. This definition assumes that there should exist some formal structure, e.g., a graph, a table with slots, a logical formula, executable program code, etc. and an algorithm that would translate a sentence in natural language into this structure. The algorithm can be based on some hand-crafted rules or trained on some data. According to Kate and Wong [2010], semantic parsing assumes deep semantic analysis, its goal is

to understand the whole sentence. Because the result of semantic parsing is meant to be read by a machine, the output is supposed to be exact and non-ambiguous.

There exist various approaches to representing the meaning of a sentence. Among the earliest meaning representation frameworks is, e.g., the one used in the LSNLIS project (the Lunar Sciences Natural Language Information System) [Woods, 1972]. The framework consists of templates that represent an extended variant of the ordinary predicate calculus notation with Boolean connectors, such as *'and'* and *'or'*, and quantifiers. The main goal of LSNLIS is to assist lunar geologists in searching necessary information in a large computer database, and the meaning representation formalism is necessary to convert the queries in natural language into a machine-readable form.

Another way to capture meaning representation is to use templates with goals (intents) and corresponding slots. Such an approach is realized, e.g., in the ATIS (Air Travel Information System) corpus created for the evaluation of the spoken language understanding systems [Price, 1990]. The corpus contains utterances and their mappings to intents and slots. The templates are designed for the domain of air travel, e.g., the intent *'Airfare'* has such slots as *'Cost_relative'*, *'Departure_city'*, *'Arrival_city'* and so on.

Speaking about more recent meaning representation frameworks, it is necessary to mention Discourse Representation Theory (DRT) [Kamp et al., 2011], which aims at dynamic interpretation of sentences depending on their context. DRT focuses on such phenomena as anaphora, conditionals, quantification, tense, presupposition, and propositional attitudes. According to DRT, a sentence/utterance evokes a mental representation in the head of its reader/hearer. This mental representation is called a Discourse Representation Structure (DRS). It can change as discourse unfolds and new information comes in. A DRS (often depicted as a box) consists of two parts: a set of discourse referents which represent objects and a set of DRS-conditions which encode the information about referents. A DRS-condition can be atomic, i.e. be a predicate, or complex, i.e. embed one or more other DRSs. More details about DRT, as well as examples, can be found in the book by Kamp et al. [2011].

The DRT framework served as a basis for the creation of a corpus called Groningen Meaning Bank (GMB) [Basile et al., 2012]. It is a semantic resource that integrates various phenomena, e.g., predicate-argument structure, scope, tense, thematic roles, rhetorical relations and presuppositions.

As Koller et al. [2019] notice, nowadays there is a growing interest in the representation of meaning in the form of directed graphs. Semantic graphs may have several roots, several incoming (except for roots) and outcoming edges, and they need not be connected, i.e. a graph may contain a pair of nodes that are not connected by a path [Oepen et al., 2019]. Among the most well-known graph-based meaning representation frameworks are, e.g., CCG word-word dependencies [Hockenmaier and Steedman, 2007], UCCA (Universal Conceptual Cognitive Annotation) [Abend and Rappoport, 2013], DMRS (Dependency Minimal Recursion Semantics) [Copestake, 2009] and AMR (Abstract Meaning Representation) [Banarescu et al., 2013]. Graph-based approaches can be classified based on different criteria. It can be the aspect of meaning they focus on, like predicate-argument structure, the interpretation of nodes, e.g., nodes may represent individual entities or more abstract objects, etc. One of the most widely used classifications relies on the relationship between the tokens of the sentence and

the nodes of the graph [Koller et al., 2019]. More information, including systematic overviews of the graph-based approaches, the interested reader can find, e.g., in papers by Koller et al. [2019]; Oepen et al. [2019]; Kate and Wong [2010].

The list of meaning representation frameworks that we have just mentioned is, of course, not exhaustive. More examples can be found, e.g., in the surveys by Kamath and Das [2018] or Zhu et al. [2019]. It should also be said that most of the frameworks we mentioned focus more on general concepts rather than on nuances of lexical meaning. However, often it is also important to take lexical semantics into account. One of the most advanced meaning representation frameworks that are based on it is called frame[1] semantics.

We will present frame semantics in detail in the next section, and only point out its differences to the rest of the approaches. First of all, frame semantics does not use any logical formalisms or graphs, instead it splits the sentence into spans (simple substrings) with respect to target token(s) and assigns categories to them like, e.g., semantic role labeling does. So, the representations are simple and can be easily read and understood by a human. Consequently, much less effort is required for the annotation of text with semantic frames and frame elements. Second, the representations capture only the lexical meaning and do not include any information about quantification, scope, tense, etc. Third, sentence representations are not supposed to be dynamic, i.e. frame semantics does not have enough inventory to track changes and update the categories respectively. So, on the one hand, frame semantics has lots of limitations, but on the other hand, its simplicity and the fact that it focuses on lexical semantics make it very attractive as a way of meaning representation, which can be combined, e.g., with the logic-based frameworks.

2.2 Frame semantics

In this section we will briefly summarize the main ideas behind the theory of frame semantics, give a concise description of the FrameNet database for the English language that has roots in this theory, present the basic frame annotation principles for text and dialogue, and introduce the most important concepts that will be used throughout the work, such as *frame*, *target*, *lexical unit*, etc., and provide examples illustrating them. We will also give an overview of other corpora annotated with semantic frames besides FrameNet, including text and dialogue data of different size and from different domains. We will compare them with the TRADR data and discuss the possibility of using them as training data for our model. Next, we will summarize the most notable frameworks that perform frame-semantic parsing, including frameworks that focus only on automatic frame assignment, and more complex ones that also perform frame element recognition and classification. In addition, we will present the main achievements in frame-semantic parsing at discourse level. To provide the reader with a systematized overview of the area, we will track the development of frame-semantic-parsing through time, outline the main tendencies and compare various frameworks with each other.

[1]Note that we use the term 'frame' as defined by Ruppenhofer et al. [2006], and not in its narrow sense meaning a structure with slots used in task-oriented dialogues.

According to Petruck [1996], frame semantics is a research program in empirical semantics which emphasizes the continuities between language and experience, and provides a framework for presenting the results of that research.

The theory of frame semantics goes back to the 1970s. One of the pioneers in this area of research was Charles J. Fillmore. He suggested that a language description should include not only lexicon and grammar, but also a set of *'frames'* that incorporate the semantics of the language elements [Fillmore, 1976]. Fillmore [1982] defines a frame as a system of concepts which are related to each other, and states that one cannot understand a separate concept without understanding the whole structure it is a part of. This structure usually comes to mind automatically, when we hear a word in a certain context. Frame semantics tries to describe and formalize such structures. The notion of frame can be traced back to Fillmore's *'case frames'*, that were parts of a framework called *'case grammar'* [Fillmore, 1967]. According to case grammar, the meaning of a sentence is conveyed by a predicating word which is a verb. Each verb is associated with a certain number of slots or *'cases'*. The elements that fill in the slots are the verb's arguments. Fillmore introduced six main argument types: *Agentive*, *Instrumental*, *Dative*, *Factitive*, *Locative* and *Objective*. Case frames are descriptions of predicates (verbs) together with semantic roles of their arguments, and represent abstract situations or scenes. The main disadvantage of case grammar was the fact that grouping verbs by their valency patterns did not always allow to capture the difference in their meanings, e.g., the difference between the phrases *"give it to John"* and *"send it to Chicago"* was lost [Fillmore, 1982]. To overcome the limitations of mixed syntactic-semantic valence description of predicates, in his subsequent research Fillmore focused more on frames rather than on verbs and arguments.

It should be emphasized that Fillmore [1982] uses the word *'frame'* as a general cover term for such concepts as 'schema', 'script', 'scenario', or 'cognitive model'. According to Fillmore, all meaningful words in a language evoke in our mind certain situations, and frames serve to identify them and give them structure and coherence. He stressed that the frames are universal, however, the way they are realized may differ from language to language [Fillmore, 1976].

The reason for this difference is the fact that frames reflect the environment, the model of the world, experiences and cultural background of a language user. The following example taken from a paper by Petruck [1996] illustrates the idea: the sentence *"Julia will open her presents after blowing out the candles and eating some cake"* evokes in our minds a birthday party scene despite the fact that the word *'birthday'* was not mentioned. We simply automatically associate *'presents'*, *'blowing out the candles'* and *'eating cake'* with a birthday party based on our cultural experience.

So, we can conclude that frame semantics aims at systematic description of lexical meanings of words by means of giving structure to abstract stereotypical events and situations evoked by these words. In the rest of the current section we will show how the ideas of frame semantics can be realized in practice.

2.2.1 FrameNet project

The theory of frame semantics served as a basis for the FrameNet project that started in 1997 [Baker et al., 1998]. The project resulted in the following achievements. First,

the notions of frame, frame element (FE) and target got clear definitions presented below.

Frame is a schematic representation of a situation involving various participants, props (i.e. any inanimate entities that figure into the description of a scene characterized in a frame) and other conceptual roles, each of which is a frame element.
Frame element is a frame-specific defined semantic role that is the basic unit of a frame.
Target is the frame-evoking element in respect to which the annotation is provided.

Second, a complex lexical database that is both human- and machine-readable was created and made publicly available online. Third, a software suite for editing and annotating frames was developed. Later there also appeared a number of APIs that allow convenient ways to work with the FrameNet database, e.g., the NLTK FrameNet API [Schneider and Wooters, 2017] and *Valencer* [Kabbach and Ribeyre, 2016].

It is necessary to give some more details about the FrameNet database. It consists of two closely related and intertwined parts.

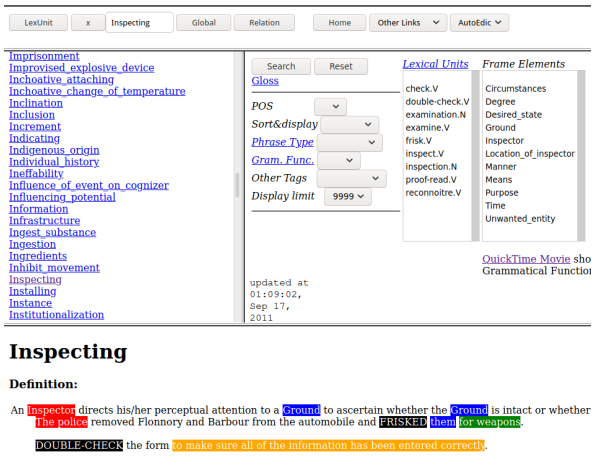
The **lexical database** contains more than 13,600 lexical entries also known as lexical units (LUs), which can be represented by a separate word, a multi-word expression or an idiomatic phrase [Ruppenhofer et al., 2006]. Each entry introduces a particular word sense that evokes a certain frame. It includes a brief sense description, a table of FEs and their syntactic representations that are typical for this word sense, possible valence patterns found in the data for this LU and a list of manually annotated examples.

The **frame database** includes more than 1,200 frames. For each frame there is a definition, a list of core (i.e. essential to the meaning of a frame), non-core (i.e. not uniquely characterizing) and extra-thematic (relating to or encompassing another frame) FEs, information about various relations connecting a frame with other frames and a list of LUs that evoke this frame. Examples 2.2.1 and 2.2.2 illustrate the definition of *'Inspecting'* frame and some of its FEs. Notice that in Example 2.2.2 one of the FEs, namely *'Unwanted_ entity'* is absent, and the abbreviation DNI is given instead. DNI stands for *'definite null instantiation'* and is used to mark the core element that is omitted, because it can be understood from discourse context. There also exist other types of null instantiation. They include INI (*indefinite null instantiation*) employed to illustrate the missing objects of transitive verbs (e.g., *eat*, *sew* or *bake*) used intransitively, and CNI (*constructional null instantiation*) which introduces grammatically licensed omissions, e.g., in imperative or passive structures.

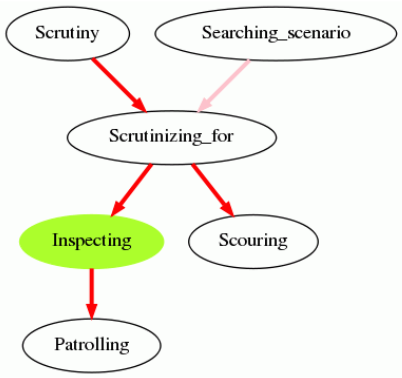
Example 2.2.1. *'Inspecting' Frame Definition*
An INSPECTOR *directs his/her perceptual attention to a* GROUND *to ascertain whether the* GROUND *is intact or whether an* UNWANTED_ENTITY *is present. Alternatively, the desired outcome of the inspection may be presented as a* PURPOSE.

Example 2.2.2. *'Inspecting' Frame's FEs*
[$_{INSPECTOR}$ *He] moved toward the control panel and [*$_{TARGET}$ ***inspected]***
[$_{GROUND}$ *it] [*$_{LOCATION\_OF\_PROTAGONIST}$ *from a distance], [*$_{MEANS}$ *without touching it] [*$_{UNWANTED\_ENTITY}$ *DNI].*

FrameNet is stored as a relational database in MySQL, accessible via a web interface called *Frame*SQL (see Figure 2.1a). The interface allows users to make queries. It is also possible to trace the relations between frames using the *FrameGrapher* (Figure 2.1b) and learn more about the FrameNet as a frame taxonomy via *FrameLatticeList* (Figure 2.1c). To explore these features, the interested reader can refer to the Internet site of the project [FrameNet, 2020]. The database can be extracted in the XML format and used for various NLP applications.



(a) *Frame*SQL user interface



(b) *FrameGrapher* shows how e.g., the frame *'Inspecting'* relates other frames: red arrows stand for *'Inheritance'*, pink ones - for *'Perspective On'* relations

Event Tree

- Event, ID# 187. Definition. (Total related: 646)
  - 631 frames related to Event via Inheritance (24 immediate Inheritance, 299 total Inheritance).
  - 17 frames related to Event via Using (7 immediate Using, 13 total Using).
    - Frequency, ID# 80. Definition. (contains 4 Using relations)
      - 5 frames related to Frequency via Using (1 immediate Using, 4 total Using).
        - Custom, ID# 380. Definition. (contains 3 Using relations)
          - 4 frames related to Custom via Using (3 immediate Using, 3 total Using).
            - Manner_of_life, ID# 2562. Definition
            - Expected_location_of_person, ID# 1619. Definition.
              - 1 frame related to Expected_location_of_person via Inheritance (1 immediate Inheritance, 1 total Inheritance).
                - Temporary_leave, ID# 2709. Definition
            - Craft, ID# 981. Definition
    - Participation, ID# 1144. Definition. (contains 1 Using relation)
    - Hindering, ID# 480. Definition. (contains 1 Using relation)
    - Manner, ID# 2563. Definition
    - Locale_by_event, ID# 1921. Definition.
    - Temporal_pattern, ID# 1040. Definition
    - Preventing, ID# 516. Definition.
  - 1 frame related to Event via Precedes (1 immediate Precedes, 1 total Precedes).

Relation Tree

- Relation, ID# 248. Definition. (Total related: 67)

(c) *FrameLatticeList* shows all frames that can be traced via different relations to one of the root frames, e.g., the *'Event'* frame

Figure 2.1: Various tools for working with the FrameNet database

2.2.2 FrameNet annotation principles

The creation of both lexical and frame databases and investigation of valence patterns of hundreds of LUs involved the annotation of thousands of sentences which are

now available as the FrameNet corpus. In this section we will talk about FrameNet annotation approaches and principles as presented in the annotation guidelines by Ruppenhofer et al. [2006] and give a couple of examples. This can help better understand the theory of frame semantics, as well as its differences from other meaning representation frameworks. We will start with two main approaches to frame annotations used in the FrameNet project.

The first one is called **lexicographic annotation**. Its main goal is to study all possible valency patterns of a LU in each of its senses, and store them in the database together with examples.

The second one is called **full-text annotation**. It assumes that each meaningful word in a sentence can evoke a frame. This results in creating several annotation sets for each sentence.

Currently, there are more than 174,500 lexicographic annotation sets and more than 28,400 full-text ones in FrameNet. The data for the lexicographic annotation mostly comes from the British National Corpus (BNC). And that for full-text annotation was drawn from several sources, such as the American National Corpus (ANC), the Nuclear Threat Initiative website, and the Wall Street Journal (WSJ) [Baker, 2008]. More details about the data sources will be given in the next chapter.

Both annotation approaches assume having three main layers of annotation for each target. The first layer consists of FEs annotation of a frame evoked by a certain target, which is annotated too. The goal is to find in the sentence all the dependents of the target word. As mentioned earlier, in case some core FEs are absent in the sentence, the empty slots must be marked with the correct null instantiation type. As a rule, not only the heads but the whole syntactic constituents of the target are annotated (see Example 2.2.2). The second layer serves to identify phrase types of the syntactic constituents. There are several kinds of noun, prepositional, verb and other phrases. Finally, the third layer is used for assigning grammatical functions to the FEs. Each target type, i.e. verb, noun, adjective, and so on, is associated with certain grammatical functions. The main ones are *'External Argument'*, *'Object'* and *'Dependent'*. The full lists of phrase types and grammatical functions can be found in [Ruppenhofer et al., 2006]. Example 2.2.3 shows how we can use them for annotation. Notice that the FEs from Example 2.2.2 presented earlier are actually assigned on top of the corresponding syntactic structures/grammatical functions demonstrated in the current example.

Example 2.2.3. *'Inspecting' Frame's phrase types and grammatical functions*
*[$_{NP.Ext}$ He] moved toward the control panel and [$_{TARGET}$ **inspected**] [$_{NP.Obj}$ it] [$_{PP[from].Dep}$ from a distance], [$_{PP[without].Dep}$ without touching it] [$_{DNI.-}$ DNI].*

We will examine the FrameNet corpus with respect to the distribution of semantic frames and the ambiguity of LUs in Chapter 3, and in Chapter 4 we will use the corpus as primary and additional data for training a frame classifier.

2.2.3 Corpora with semantic frame annotation

Besides the FrameNet data, there exist other corpora annotated with semantic frames. Here we will present those corpora that we are aware of. They include text and dialogue

data, and can be either freely found online or requested. Their overview will be given in Table 2.1.

To start with, there exists the Manually Annotated Sub-Corpus (MASC) [Ide, 2017]. It is in English. The corpus represents a part of the Open American National Corpus (OANC) and consists of 500,000 tokens. The data belongs to 19 different genres and contains 20 various types of annotation including semantic frames. However, dialogue data makes merely 15% of the corpus and only about 40,000 tokens of the corpus were annotated with frames using the full-text annotation approach [Passonneau et al., 2012]. The corpus is in the TSV format and freely available.

A dataset in English called 'Yahoo! Answers Gold Standard' (YAGS) was created specifically for testing frame-semantic parsers [Hartmann et al., 2017]. It is based on web user-generated questions and answers and contains more than 1,400 sentences, 3,000 frame annotations, and 6,000 semantic role annotations. The corpus is in the XML format and can be freely used for research purposes. The annotations are stored separately as simple text files.

One more interesting dataset is FN-RE [Alhoshan et al., 2018]. It contains 220 user requirements (about 1,148 sentences with 21,012 tokens) meant for software development, which were annotated with semantic frames and frame elements. The data comes from various online sources, is stored in the XML format and is publicly available. The corpus is also in English.

Another dataset containing annotations of frames, targets, frame elements and syntactic roles is called SALSA [Burchardt et al., 2006]. It is a German newspaper corpus. It includes annotations of more than 35,000 verbal and nominal instances. Altogether the corpus makes up 838,307 tokens. It is in the TIGER XML format and can be requested for research.

There exist very few purely dialogue corpora annotated with semantic frames, and nothing that we were able to find is in English.

One of such corpora is, e.g., an Arabic TIHR_ARC corpus of 4,000 dialogues, collected using the *Wizard of Oz* approach, when a human operator simulates the responses of a telephone server. The corpus belongs to the touristic domain and was annotated with semantic frames as part of the project on improvement of human-machine dialogue systems [Lhioui et al., 2017].

A corpus called RATP-DECODA in French contains more than 2,000 manually transcribed telephone conversations of the Customer Care Service [Lailler et al., 2016]. The topics include schedules, traffic states, lost and found objects, railway related problems and many others. The corpus contains more than 146,000 frame annotations based on the guidelines provided in the paper by Trione et al. [2015].

Another dialogue corpus annotated with semantic frames was created in the course of the LUNA (Language Understanding in multilingual communication systems) project. This corpus is in Italian. It contain 180 human-human (9,074 utterances with 66,290 tokens) and 249 human-machine (1,525 utterances with 12,420 tokens) dialogues, recorded in the call center of the customer service of the Italian Consortium for Information Systems. The former are spontaneous conversations about software/hardware problems, the latter were recorded under the Wizard of Oz setting using ten predefined scenarios [Raymond et al., 2008].

As our task is to perform automatic frame assignment in team communication

in disaster response scenario, and the corpus we have (TRADR) is rather small, we were very interested in finding other dialogue corpora annotated with semantic frames from the same or related domain. However, we see that the absolute majority of the presented dialogue data is not in English or German, which makes it difficult to use them (multi- or cross-lingual approaches can be a part of future research). Next, all the available text corpora in English or German come from the domains that are very different from ours, and most of these corpora are not very large either. Finally, each corpus is organized and structured in its own way (see Table 2.1).

Corpus	Lang.	Form	Domain	Size	Format	Publications
FrameNet	English	text	newspaper texts	199,508 sentences 4,751,140 tokens	XML	FrameNet [2020]
MASC	English	text & dialogue	various genres	# sentences n/a 500,000 tokens	GrAF	Ide [2017]
YAGS	English	text	question-answer pairs from Yahoo! Answers	1,415 sentences # tokens n/a	n/a	Hartmann et al. [2017]
FN-RE	English	text	user requirements for software development	1,148 sentences 21,012 tokens	XML	Alhoshan et al. [2018]
SALSA	German	text	newspaper texts	35,236 sentences 838,307 tokens	TIGER XML	Burchardt et al. [2006]
TIHR_ARC	Arabic	dialogue	tourism	4,000 dialogues # sentences n/a # tokens n/a	n/a	Lhioui et al. [2017]
RATP-DECODA	French	dialogue	customer care service	2,109 dialogues # sentences n/a # tokens n/a	XML	Lailler et al. [2016]
LUNA	Italian	dialogue	customer care service	10,599 utterances 78,710 tokens	TIGER XML	Raymond et al. [2008]

Table 2.1: Overview of corpora annotated with semantic frames ('n/a' stands for 'not available')

As none of the presented datasets is ideally suited for our purposes, we will focus on the TRADR corpus as our primary data, and use the SALSA corpus as a source of additional training data for our German frame classifier. For training the English version of the classifier we will use the FrameNet data. In contrast to other datasets both FrameNet and SALSA are quite large and have APIs allowing to work with the data conveniently. We will have a closer look at the SALSA corpus together with FrameNet and TRADR in Chapter 3.

To wrap up our summary of the available corpora annotated with frames, we would like to mention that most of them, including dialogue data, were annotated following the FrameNet annotation guidelines (see Ruppenhofer et al. [2006]). We used these guidelines to annotate the TRADR data too, as it was difficult to find any instructions designed specifically for dialogue. One of such rare examples is the LUNA project. Their annotation instructions were very helpful to us. We relied on them to deal with incomplete/elliptical and ungrammatical utterances, figurative meaning and new senses that are not part of the FrameNet database. More details about the annotation of the TRADR corpus with frames can be found in Appendix A.

2.2.4 Frame semantics in NLP

In what follows we will present the most notable practical applications of frame semantics in NLP. Earlier, in Section 2.1 we noticed that frame semantics is not one of the most common meaning representation frameworks, however, since 2007, when frame semantic structure extraction got included into the list of SemEval'07 tasks [Baker et al., 2007], this research area has been gradually growing, and there regularly appear new papers devoted to the topic of frame-semantic parsing. Most of them describe approaches aiming to work on text data, but the outcomes of this research still can be useful to us.

Some of the projects deal only with automatic frame assignment, others have a bigger goal, namely, recognizing targets, frames and frame elements. And whereas most frameworks focus on individual sentences, there also exist approaches to frame-semantic parsing at discourse level.

We start with examples of the frameworks that concentrate on automatic **frame assignment**. This task assumes that the target is already given, and the main challenge is to choose the correct frame in case of polysemous targets.

One of such frameworks is *SimpleFrameId* from the creators of the YAGS dataset [Hartmann et al., 2017]. The main idea behind this framework is to represent the target and its context as vectors, concatenate them and train a classifier based on neural networks that will assign the best-scoring frame to the given target-context representation. As an option, it is possible to perform frame filtering, using the information about mapping of certain lexical units (LUs) to certain frames from the FrameNet database.

The framework TSABCNN described in the paper by Zhao et al. [2018] also assumes that the target is already identified and focuses on frame assignment. Each word in a sentence is represented using *word2vec* embeddings. These representations are concatenated with position vectors capturing the positions of words with respect to the target. A classifier based on convolutional neural networks is then trained to learn the frame labels.

The most recent approaches to frame assignment rely on BERT embeddings and pretrained models [Devlin et al., 2019]. E.g., Sikos and Padó [2019] present four different models based on BERT embeddings, with and without fine-tuning.

A framework called PAFIBERT [Tan and Na, 2019] is also based on BERT embeddings and on a pretrained BERT model. Like *SimpleFrameId*, PAFIBERT makes use of context information and allows to do frame filtering. As part of the fine-tuning process it uses an attention mechanism to give weights to words that make up the context of the target.

One more interesting approach to automatic frame assignment was introduced by Ribeiro et al. [2020]. It focuses on verbal frame-evoking targets and represents them using contextualized ELMo embeddings. In case the target is multi-word, Ribeiro et al. [2020] use dependency parsing to get the head word. Next, the targets are treated as nodes in a graph. A pair of nodes is connected only if the distance between them is smaller than a certain threshold. The nodes are clustered using *Chinese Whispers* algorithm [Biemann, 2006]. A new instance is classified by determining the closest cluster. The approach is fully unsupervised.

We must emphasise that all the systems briefly described above were trained on

text and never tested on dialogue data.

Now we will present the frameworks that perform **frame-semantic parsing**, i.e. they detect frame-evoking targets, frames, argument (frame element) spans, and classify frame elements. We will start with the frameworks developed for text data.

The approach to frame elements identification suggested by Gildea and Jurafsky [2002] is considered to be a pioneering work in the area of frame-semantic parsing, despite the fact that such typical steps as target and frame detection are omitted. To identify frame element labels Gildea and Jurafsky [2002] train a statistical classifier. First, they parse the training data and manually match the frame elements with the corresponding tree constituents. Second, they extract various features, such as phrase type, governing category, position, head word and so on to represent these constituents. The classifier learns the probabilities of semantic roles given a set of features. Gildea and Jurafsky [2002] also examine the problem of frame element boundary detection. They train a separate binary statistical classifier that learns to predict for a syntactic tree constituent given as a set of features, whether it is a frame element or not.

One of the first frameworks that was built to identify targets, frames and frame elements is called LTH [Johansson and Nugues, 2007]. It performs target identification using a set of hand-crafted rules. The frame and frame element recognition steps employ dependency syntax and SVM classifiers. For non-ambiguous targets a frame can be retrieved using a simple mapping. Ambiguous targets are represented as feature vectors, and include such features as target's lemma, a set of target's dependencies and so on. An SVM classifier is trained to disambiguate them. Frame element identification is realized as a two-step procedure: argument identification and argument classification. Both steps imply filtering out some nodes, e.g., support verbs and copulas, from the parse tree and training an SVM classifier based on hand-crafted features, such as the target lemma, voice, or dependency path from the target to the node.

Das et al. [2010] with a model called SEMAFOR follow the approach presented by Johansson and Nugues [2007]. They also rely on a set of rules for automatic target extraction and use a very elaborated set of features to represent targets and frame elements, but replace SVM classifiers with feature-based, discriminative probabilistic (log-linear) models. In addition, Das et al. [2010] examine a problem of overlapping frame elements and apply beam-search technique to avoid this.

One of the most well-known frameworks was suggested by Hermann et al. [2014] and also performs frame and argument identification given a target. It is one of the first semantic parsers to use embeddings. The target is represented as a vector, certain parts of which are reserved for certain argument representations. If some argument is missing, its slot is filled with zeros. Then this sparse, high-dimensional vector is mapped to a low-dimensional space. All frame labels are also represented in this low-dimensional space as vectors. The classifier learns to minimize the distance between the target vector and the correct label vector. For argument identification Hermann et al. [2014] use a set of semantic roles of the given frame and a rule-based argument extraction algorithm that in its turn exploits the dependency tree of the sentence. The procedure is very similar to that of Das et al. [2010].

An interesting approach to frame-semantic parsing is realized in a framework named Open-Sesame [Swayamdipta et al., 2017]. It is one of the first frame-semantic parsers employing neural networks and embeddings for all the steps in the pipeline, namely

frame, argument span and argument identification. The model uses bidirectional LSTM for frame identification and segmental RNN based on combination of biLSTM with a semi-Markov conditional random field (CRF) for span and argument detection, and does not require syntactic or dependency parsing. However, the authors of the framework mention that adding information about the syntactic structure of the sentence helps to improve the performance of the classifier.

Yang and Mitchell [2017] present another framework that performs frame and argument identification with the help of neural networks. Frame identification is done using a simple multi-layer network, while the second task assumes more complex approach, namely training two different neural networks. The first is called sequential and relies on biLSTM with CRF and aims at modeling sentence-level information. The second is called relational. It is a simple multi-layer network which captures span-level dependencies between the target and its arguments. The second model is trained using the knowledge extracted from the first one.

One of the most recent approaches to frame-semantic parsing was presented by Kalyanpur et al. [2020]. It is interpreted as sequence-to-sequence generation problem, and is based on the encoder-decoder architecture, namely on the T5 model, which is available via the *HuggingFace* library [Wolf et al., 2020]. In contrast to other frameworks mentioned earlier, Kalyanpur et al. [2020] do not treat frame-semantic parsing as a pipeline, where frame identification is followed by argument detection and classification. Instead, they employ multi-task learning which assumes that two or more models learn their tasks separately, but share parameters or part of the architecture. Another notable characteristic of the approach is the use of New Oxford American Dictionary (NOAD) [Lindberg and Stevenson, 1999] to provide a broader coverage as the FrameNet word senses are not exhaustive.

Next, we will present two frame-semantic parsing frameworks designed for dialogue. Unfortunately, they are not very recent, but still of a special interest to us.

As part of the above mentioned LUNA project, Coppola et al. [2008] developed a frame-semantic parser for conversational speech. The parsing process includes four main steps: target detection, frame disambiguation, frame element boundary detection, semantic role classification. The first two steps are performed using heuristics and a multi-classification approach, the realization of the last two steps is very similar to the LTH framework and involves syntactic parsing and training SVM classifiers. The difference is that Coppola et al. [2008] use additional structural features for SVM. They are tree kernels, polynomial kernels and their sum. Tree kernels are scalar products that evaluate the number of common subtrees between pairs of parse trees. A polynomial kernel is also a scalar that measures the strength of the relationship between two objects: $PK(o_1, o_2) = (c + \vec{x}_1 \cdot \vec{x}_2)^d$, where o_1, o_2 are objects, c - constant, $\vec{x}_1, \vec{x}_2$ - feature vectors and d - degree of the polynomial.

A semi-supervised approach to frame and frame element identification in dialogue for French is presented by Trione et al. [2015]. Its main goal is actually to speed up the manual annotation process, not pure frame-semantic parsing. The framework includes three levels: data preprocessing, syntactic parsing adapted for conversational speech and semantic parsing. The focus is on frame and argument span identification. The targets and frames are detected with the help of a hand-crafted set of lexical triggers, which includes 200 most frequent words from 7 domains. To perform span detection,

Trione et al. [2015] adjust a dependency parser to the processing of spontaneous speech. This is an iterative process: parse errors are analyzed and manually corrected, the parser is retrained, and the procedure is repeated until the parses are good enough. This is a very costly process. Frame element classification is performed manually.

Characteristic features	Framework											
	Gildea & Jurafsky (2002)	LTH (2007)	LUNA (2008)	SEMAFOR (2010)	Hermann et al. (2014)	SimpleFrameId (2017)	Open-Sesame (2017)	Yang & Mitchell (2017)	TSABCNN (2018)	PAFIBERT (2019)	Ribeiro et al. (2020)	Kalyanpur et al. (2020)
hand-crafted rules		✓	✓	✓	✓							
hand-crafted features	✓	✓	✓	✓	✓							
kernels			✓									
parsing	✓	✓	✓	✓	✓	✓					✓	
embeddings					✓	✓	✓	✓	✓	✓	✓	✓
statistical classifier	✓											
SVM		✓	✓									
conditional log-linear model				✓	✓							
neural network						✓	✓	✓	✓	✓		✓
CRF							✓	✓				
clustering	✓										✓	
graph structure											✓	
Frame assignment accuracy	n/a	62.1*	n/a	61.44*	88.41	87.63	70.7*	88.2	89.72	88.97	80.26*	90.0

Table 2.2: Comparison of various frame-semantic parsing frameworks; scores marked with '*' stand for F-score (the authors do not report accuracy); 'n/a' means that the scores are not available, because the frame recognition step is omitted

To summarize our survey of frame-semantic parsing, we would like to make the following conclusions. As for target identification, most of the frameworks assume that it is already given. If not, it is detected using heuristics and hand-crafted rules. For frame identification it is crucial to use context and target position within the sentence. Among all the parts of the frame-semantic parsing pipeline frame element detection is the most difficult and complex one. Our overview of the frameworks allows to outline several important tendencies. First, token/context representation has changed from feature-based vectors to embeddings (often pretrained ones). As for argument span representations, syntactic/dependency trees also got replaced with embeddings. There happened a shift from rule-based and statistical models to neural network approaches, and gradually complex neural network frameworks, e.g., TSABCNN or Open-Sesame, gave way to simpler and more efficient ones based on pretrained embeddings and models like BERT. Table 2.2 compares most of the frameworks we have discussed, and is a good illustration of tendencies that have been mentioned above. The last row of the table presents the best results that these frameworks were able to achieve for the task of automatic frame assignment.

So far we have discussed the frameworks that aim at capturing the meaning of isolated sentences. However, for many practical applications it is more beneficial to

examine a sentence (or an utterance) as part of discourse and establish cross-sentence links between the semantic structures, e.g., resolve implicit frame elements (the so-called *null instantiations* in FrameNet) and find co-referential arguments.

One of the first attempts to analyse FrameNet as a means of knowledge representation at discourse level was made by Fillmore and Baker [2001] who researched FrameNet's potential application for the task of text understanding. They view FrameNet as a way to assist with word-sense disambiguation, find semantic dependents of frame-evoking elements, choose valence-justified parses, and activate topic-related vocabulary for recognition and selection of the correct sense in successive parts of a text. As an example, the researchers use FrameNet for the analysis of a small newspaper text. They rely on syntactic parsing to identify frames and frame elements in all the sentences, and try to determine parts of the structures that make the sentences coherent. However, Fillmore and Baker [2001] focus only on the analysis and do not suggest any algorithmic approaches.

A similar study was performed by Burchardt et al. [2005]. The researchers also analyse a piece of newspaper writing. They assign frames and frame elements to it, investigate different types of relations between frames and arguments and try to find characteristic patterns in these relations which would allow to infer co-reference relations between frame elements and resolve implicit arguments. Burchardt et al. [2005] differentiate between contextual relations that are based on deep parsing, e.g., structural embedding, adjacency or discourse relations, and lexico-semantic frame relations captured by FrameNet, such as *Inheritance* and *Subframe* relations. The authors also mention that further inference of co-reference relations and resolution of implicit arguments can only be heuristic, as FrameNet provides partial conceptual structure. Finally, the researchers present some ideas how the process of inference can be generalized and automatized using the LFG grammars. More information about LFG grammars can be found in the work by Butt et al. [2004].

Ruppenhofer et al. [2010] drew more attention to the topic of frame-semantic parsing beyond the sentence-level, when they included the problem of linking events and their participants in discourse into SemEval'10 tasks. The task focused on frame element classification and on finding and filling implicit arguments. Due to the difficulty and novelty of the task, only a few frameworks were submitted. In what follows, we will briefly introduce two of them (the third framework is omitted as it performs frame element resolution only at sentence level). As mentioned in Section 2.2.1, FrameNet distinguishes between three types of implicit arguments: constructional, indefinite and definite null instantiations. Both systems deal only with the third type of null instantiations.

The first framework was already presented earlier. It is SEMAFOR [Das et al., 2010] which was adapted for implicit argument recognition and resolution. The framework models implicit arguments linking as a variant of role recognition with the set of potential arguments being extended with noun phrases from the previous sentence. The system also uses, among other information, semantic similarity between the heads of potential arguments and role fillers in the training data.

The second framework called VENSES++ was developed by Tonelli and Delmonte [2010]. It is a reduced version of their other framework called GETARUNS designed for text understanding. VENSES++ is based on the LFG theory and has three modules:

lexico-semantic, anaphora resolution and deep semantic. The modules are joined into a pipeline which given a sentence outputs a predicate-argument structure as a logical form. The system uses its own set of arguments that only partially overlap with FrameNet frame elements. So, VENSES++ output is first mapped to frame elements. Next, if the predicate is a verb, the system decides which core elements are missing by using some hand-crafted rules and comparing the output with valence patterns of lexical units available via the FrameNet database. To find candidates to fill in the empty slots, VENSES++ checks previous sentences for semantically close predicates. In case of a nominal predicate, the system first goes through the so-called History List, where all the nominal heads from previous sentences are stored, and extracts heads that are semantically related to the predicate. Then, using the FrameNet database, it checks what core frame elements are missing and tries to find the right candidates among the extracted heads with the help of hand-crafted rules.

Since 2010, we have observed some further notable approaches to the problem of implicit arguments resolution using context. One of them was presented by Laparra and Rigau [2012]. They associate all frame elements (their heads, to be exact) in training data with semantic types, which are basically sets of features, such as WordNet sense, ontological features, lemma, etc. Next, they train a model that learns a probability distribution of semantic types of each frame element. Now, to track all DNIs, Laparra and Rigau [2012] count frequencies of frame element pattern for all lexical units in training data. So, given a certain lexical unit, all frame elements of the most frequent pattern are considered as potential DNIs. The system looks for the most likely candidates using a window of three previous sentences.

Roth and Frank [2015] presented a framework that also aims at inducing implicit arguments from pairs of comparable texts. The texts are newspaper articles and were grouped by matching headlines. The framework consists of two steps. The first step assumes aligning predicate-argument structures across the text pairs. The structures are represented as bipartite graphs, where predicates are nodes connected by a weighted edges. The weight defines how similar each two predicates are. It is deduced using arguments, contexts and other features. The graphs get recursively split into subgraphs, and similar predicates get aligned via clustering. In the second step the alignments are used together with heuristic rules to infer implicit arguments.

Unfortunately, we were not able to find any more recent approaches to frame-semantic parsing at discourse level. Actually, the topics of semantic role labeling and establishing co-reference between the arguments are still actively researched, but many contemporary researchers focus on PropBank arguments instead of FrameNet. One of the obvious reasons for this is that the set of frame elements in FrameNet is semantically much more fine-grained than the set of PropBank arguments. This, as well as the fact that FrameNet elements are not systematized into any taxonomy (at least, officially), makes them difficult to use for the retrieval of implicit arguments or establishing relations across sentences in a piece of discourse.

To conclude our overview of frame-semantic parsing, we would like to emphasize that even though in the current thesis we will focus only on automatic frame assignment, we found it important also to present the current state of research in the field of frame-semantic parsing in general, and to summarize and compare the most notable frameworks designed to perform other steps of frame-semantic parsing, such as frame

elements recognition and classification. This can help the reader acquire a better understanding of how the ideas of frame semantics can be realized in practice. In addition, our overview can be relevant for future research. In our work we will use the PAFIBERT model with position-based attention mechanism by Tan and Na [2019]. It is considered to be one of the state-of-the-art frame classifiers (the model showed the accuracy of about 89% when evaluated on in-domain test set), and it is easier to re-implement in comparison with other neural networks based classifiers that we mentioned. We will present our experiments with the PAFIBERT model in Chapter 4.

So, in this section we have presented the theory of frame semantics, its realization in the FrameNet project, several text and dialogue corpora annotated with semantic frames, and a number of frameworks designed to perform frame-semantic parsing. We have compared the available corpora with TRADR data, discussed which ones can be used for our work and explained our choice of FrameNet and SALSA corpora. We have also introduced various approaches to frame semantic parsing and came to conclusion that the PAFIBERT model by Tan and Na [2019] looks the most suitable for our purposes.

2.3 Multiclass classification with neural networks

To train a classifier that would be able to differentiate between hundreds of classes (e.g., in FrameNet there are 1,024 different frame labels) is not a straightforward task. There exist various approaches. Originally, most of them were designed for binary classification problems. Following Aly [2005], they can be divided into three main groups: algorithms that assume a simple extension from binary, algorithms based on transformation to binary and hierarchical classification.

In this section we will focus on multiclass classification with neural networks, as all our models that will be presented in Chapter 4 are based on them. Neural networks belong to the group of binary approaches that can be easily extended to work with multiclass data. Other algorithms from this group include decision trees, k-Nearest Neighbor, Naive Bayes and Support Vector Machines. More information about them, as well as a summary of the algorithms that are based on transformation to binary and hierarchical classification can be found in Appendix C.

We will start the section with an explanation of what a very simple feedforward neural network may look like and how it functions. Next, we will discuss what a transformer block is, and how it was realized in BERT which can be considered one of the most successful models for many NLP tasks. In Chapter 4 we will see how BERT can be used for the task of automatic frame assignment.

2.3.1 Neural Networks

A feedforward neural network usually consists of an input layer, an output layer and one or more hidden layers in between. Each hidden layer is made up of the so-called 'neurons' and is followed by an activation function. The output layer also has an activation function. This function makes the neurons non-linear, scaled and differentiable [Sun et al., 2009]. In other words, its purpose is to 'switch on' the important neurons and 'switch off' the ones bearing unnecessary or redundant information. The hidden

layers are connected via weight matrices. In the beginning the weights are initialized to very small random numbers.

If the network is used to train a binary model, then the output layer contains only one neuron, and the result of the activation function can be interpreted as a probability with which an input instance belongs to the positive class. Depending on this probability, the instance gets a label 1 (positive class) or 0 (negative class). The label is then compared with the gold label using a loss function, and the weight matrices are updated via a backpropagation process [Sun et al., 2009].

The training process is iterative, i.e. the calculations are repeated for every new training instance/batch for several epochs until the weights are adjusted and the difference between the output and gold labels is minimized.

We see now that a binary network can be easily transformed to a multiclass one by adding a certain number of neurons to the output layer. After the application of the activation function, each separate neuron is mapped to either 1 or 0 based on some threshold. How many neurons to add depends on how the labels are encoded.

One-per-class-coding assumes that the labels are one-hot vectors, such that each vector contains a single high (1) bit at a certain position with all the remaining elements being low (0) bits, as shown in Figure 2.2a [Wikipedia, 2020f]. If we have K classes, then each one-hot encoding should be of size K, and the number of neurons in the output layer N will also be equal K. In this case the class of the input example will be defined judging from the position of the high bit in the output sequence. Notice that all the remaining neurons in the sequence should be low bits [Aly, 2005].

Distributed output coding means that the labels are interpreted as sequences of binary features, i.e. they are vectors that may contain more than one high bit, as in Figure 2.2b. Often the length of a sequence representing a label is bigger than the number of classes K. Normally, the labels have length up to $2^N - 1$, where N is the size of the output layer. Given a new instance, the classifier outputs a vector of probabilities, which are mapped either to 1 (feature is present) or 0 (feature is absent). The output is compared to all possible class labels using the Hamming distance, and the closest one is chosen [Aly, 2005].

Class 1	1000
Class 2	0100
Class 3	0010
Class 4	0001

(a) One-per-class-coding

Class 1	00000
Class 2	00111
Class 3	11001
Class 4	11110

(b) Distributed output coding

Figure 2.2: Label encodings [Aly, 2005]

It is important to point out that a network with a standard backpropagation algorithm can be very sensitive to imbalanced classes. Sun et al. [2009] write that both empirical and theoretical studies show that during the training of such a network the net error for samples of major class(es) decreases, while the net error for samples of minor class(es) grows. They explain such behavior by the fact that the weights of the network are updated in the direction of the joint gradient vector, which is dominated by the gradient vector of the major class.

2.3.2 BERT model

As all the classifiers that we are going to present in Chapter 4 rely on the pretrained BERT model, it is necessary to introduce its main components and explain how they function.

According to Devlin et al. [2019], BERT stands for Bidirectional Encoder Representations from Transformers. It is a language model that learns contextual token representations (embeddings) from the *BooksCorpus* (800M words) [Zhu et al., 2015] and English *Wikipedia* (2,500M words). BERT was trained with two objectives. One is predicting randomly masked tokens based only on their contexts, the other is a next sentence prediction, i.e. producing a score, which for each pair of sentences says how likely it is that one sentence follows the other.

BERT can be used either for extracting contextual embeddings for the given input, or for fine-tuning for a variety of NLP tasks. As of 2019, simple fine-tuning of the BERT model allowed to achieve state-of-the-art performance for 11 tasks, such as question answering, sentiment classification, entailment recognition and others (see Devlin et al. [2019] for details).

A BERT model represents a stack of transformer blocks. There are two main BERT configurations. $\text{BERT}_{\text{BASE}}$ has 12 transformer blocks, a hidden layer of size 768 and 12 self-attention heads in each block. $\text{BERT}_{\text{LARGE}}$ consists of 24 transformer blocks, has a hidden layer of size 1,024 and 16 self-attention heads in each block. All our models, as well as the PAFIBERT model rely on the $\text{BERT}_{\text{BASE}}$ model. In what follows we will illustrate what a transformer block in BERT looks like, and how it learns contextual embeddings.

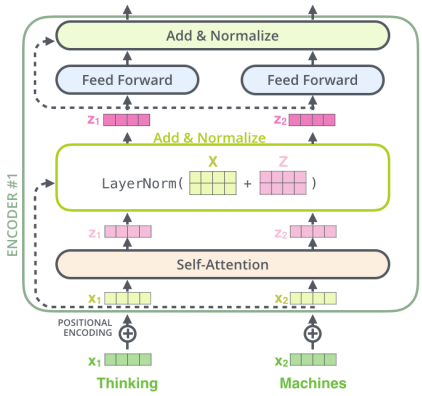


Figure 2.3: Transformer (encoder) block in BERT (taken from Alammar [2018])

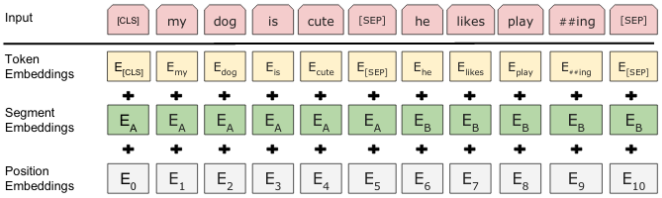


Figure 2.4: BERT input representation (taken from Devlin et al. [2019])

Figure 2.3 illustrates a single transformer (encoder) block in BERT. It consists of a self-attention layer and a feed forward layer, both followed by layer normalization. The feed forward layer is applied independently to each vector, and a residual connection (dashed line) is added around each layer, right before the normalization. In what follows we will present the components of the block in more detail.

According to Devlin et al. [2019], **BERT input** can be an arbitrary span of contiguous text, not necessarily an actual linguistic sentence. The input sequence may consist of either one or two sentences (spans) separated by a special token [SEP]. The input is tokenized using *WordPiece* embeddings [Wu et al., 2016] and 30,000 token vocabulary. Following the *WordPiece* approach, any out-of-vocabulary word is split into smaller subtokens. Another special token [CLS] is added to the beginning of the sequence. Its purpose is to aggregate syntactic and semantic characteristics of the input for further classification tasks (fine-tuning). As a result of tokenization, the tokens are mapped to input IDs, which in their turn are mapped to token embeddings. The length of the input sequence in $\text{BERT}_{\text{BASE}}$ is limited by 512 tokens.

By its nature, a single transformer block, as well as the whole network is not sensitive to the word order in the input sequence, i.e. if we shuffle up the input, we get the exact same output, whatever weights we learn [Bloem, 2019]. However, for many languages and NLP tasks the word order is important. One of the ways to capture it is to map a position of each token in the sequence to its positional embedding. Note that in BERT the positional embeddings are learnt during the network training, while the original Transformer [Vaswani et al., 2017] uses the positional encodings, which are computed using a special function that maps positions to real valued vectors without any learning.

To differentiate between two sentences in the input, the tokens in each sentence are also mapped to one of two IDs (usually 0 and 1) saying to which sentence a token belongs. For each ID a special segment embedding is learned.

As Figure 2.4 demonstrates, the input for BERT transformer network is a sum of token, segment and position embeddings.

Self-attention is the only layer in BERT that learns how the input tokens relate to each other. As Vig [2019] writes, attention is a way for a model to assign weights to tokens based on their importance to some task. It can be viewed as a function (see Figure 2.5) that given a sequence of embeddings X produces a new representation for each embedding by taking a weighted average of the original input. The attention weights depend on the current token and show how much this token attends to other tokens in the sequence. Note that the length of the new representations (denoted as Y in Figure 2.5) can be different from the length of the original embeddings, because attention usually implies linear transformations of the embeddings in X.

In order to explain what is happening inside the attention function, we will rely on the materials by Bloem [2019] and Alammar [2018]. First, it is important to note that every input embedding x_i is used in three different ways in the self attention operation:

- As a *query* it is compared to every other embedding to establish the weights for its own output y_i.
- As a *key* it is compared to every other embedding to establish the weights for the output of all the other embeddings y_j, where $j \neq i$.
- As a *value* it is used as part of the weighted sum to compute each output vector once the weights have been established.

To extract the query, key and value vectors for an embedding x_i, it is multiplied with three different trainable matrices W_q, W_k and W_v all of the same size like this: $q_i =$

$W_q x_i$, $k_i = W_k x_i$, $v_i = W_v x_i$. We assume that all three vectors have the same dimension d. Next, to know how much an embedding x_i attends to some other embedding x_j, we simply calculate the dot product between the corresponding query and key vectors: $w_{ij} = q_i^T k_j$. Figure 2.6 illustrates this procedure. Here it is calculated how much attention the embedding x_2 gives to its neighbour, the embedding x_3. The attention weight w_{23} is a dot product between the query vector (red) of x_2 and the key vector (blue) of x_3. To get a weighted representation of x_3, its value vector (green) is scaled by w_{23}. The same operations are performed between x_2 and all the other embeddings in the input, including x_2 itself. The results are summed up to get the final output y_2. The rest of the output is calculated in a similar manner.

It is necessary to mention that before the value vectors of the sequence can be scaled by the corresponding attention weights, a couple of additional operations are applied to these weights. First, the weights are scaled by $\sqrt{d}$, where d is the dimension of the query, key and value vectors mentioned earlier. Note that d is usually smaller than the dimension of input embeddings h. This is done to get rid of very large values, as they can kill the gradient, slow down learning and influence the second operation, namely *Softmax*, in a negative way. *Softmax* makes sure that all weights are positive and sum up to one.

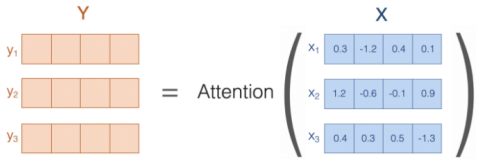


Figure 2.5: Attention as a function (taken from Vig [2019])

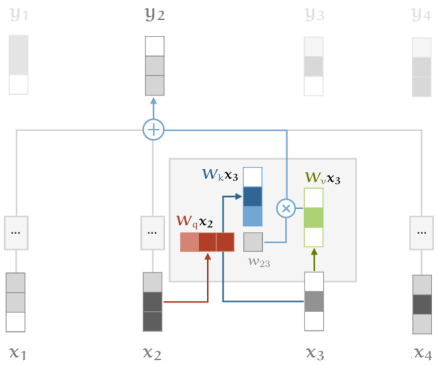


Figure 2.6: Calculating attention weights (taken from Bloem [2019])

Earlier we said that $\text{BERT}_{\text{BASE}}$ has 12 attention heads. This means that the actual attention implementation in BERT assumes building for each embedding x_i in X 12 sets of query, key and value matrices. As Alammar [2018] writes, this allows to expand the model's ability to focus on different tokens in the input and to capture a wider range of relations. Moreover, as all matrices in each attention head are randomly initialized and independent from matrices in other heads, the attention layer gets multiple representation subspaces instead of having all information summed up in one place.

So, all the attention related operations described above are performed separately for each attention head. The result of this procedure is 12 different representations of the output vector y_i. Each representation is of length d. Because the next layer in the encoder block, namely the feed forward layer, does not expect to deal with 12 different representations for each input embedding, they need to be condensed into a single vector that would incorporate the information from all attention heads. This is done by concatenating the outputs of the attention heads and multiplying the resulting

vector by an additional trainable matrix W_o. The output of this operation is then sent to the feed forward layer.

The feedforward layer expects as input a batch of sequences of embeddings of size $l \times h$, where l is the sequence length and h is the hidden layer size of 768. It consists of two simple linear layers with a $ReLU$ activation function between them. The output of this layer has the same dimensions as the input [Rush et al., 2018].

As shown in Figure 2.3, both the self-attention and feed forward layers are followed by a **layer normalization**. The layer normalization was introduced by Ba et al. [2016]. It is one of the many normalization approaches that are used in deep learning.

The main motivation behind the layer normalization is to enable smoother gradients, faster training, and better generalization accuracy [Xu et al., 2019; Kurita, 2018a]. Its difference from some other possible normalization methods lies in the dimension, along which the values are normalized. Assume that our input has form $b \times l \times h$, where b stands for batch size, l denotes sequence length and h is the size of hidden layer. Layer normalization suggests normalizing the values along the last dimension h.

So, according to Xu et al. [2019], if we take a single feature sequence $x = (x_1, ..., x_h)$, its normalized representation would be calculated as $LN(x) = g \cdot N(x) + b$, where g and b are learnable parameters that aim at scaling and increasing the magnitude of the activations, and $N(x) = \frac{x-\mu}{\sigma}$ stands for usual input normalization using mean $\mu = \frac{1}{h}\sum_{i=1}^{h} x_i$ and standard deviation $\sigma = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(x_i - \mu)^2}$ of data. More details about the layer normalization can be found in the original paper by Ba et al. [2016].

The main advantage of the layer normalization is that the statistics μ and σ are calculated independently for each feature representation and do not depend on the batch size [Kurita, 2018b].

In Figure 2.3 we can notice that before the layer normalization operation is applied, the output of the self-attention layer is summed up with its input. This is called a **residual connection**, it is depicted as a dashed line in Figure 2.3.

As Sahoo [2018] writes, adding residual connections to the network makes sense if the network consists of several blocks or layers. When the network is deep enough, due to vanishing/exploding gradients, it may have trouble learning simple functions like identity. Moreover, such networks may experience the accuracy degradation problem, when at some point the accuracy saturates and starts to decrease. Residual connections between blocks help tackle these issues.

In order to understand how a residual connection R works, it is convenient to interpret it as a difference between output and input, where the output is depicted as a function of input H: $R(x) = y - x = H(x) - x$. It is possible to rearrange this equation as $H(x) = R(x) + x$, i.e. to treat the desirable (or true) output as a sum of the output of the residual function and the input [Sahoo, 2018]. In case of BERT the residual function is represented by the self-attention layer. It is exactly what we can observe in Figure 2.3 as input for layer normalization.

To conclude this description of the components of BERT, we would like to emphasize that understanding of how BERT functions is important not only because it is the main component of all our models, but also because it can be useful for the correct analysis of these models' performance, e.g., we will later refer to BERT's attention layer in order to explain some phenomena observed during our experiments.

2.4 Handling imbalanced data

Many classification learning systems are designed to show best performance if they are trained on balanced data [Sun et al., 2009; Alam et al., 2019]. If a class is under-represented in training data, it is difficult for the model to learn it, and as result the classifier will be biased towards more frequent classes, i.e. instances that belong to a minority class will be misclassified more often [Sun et al., 2009]. This is something one would like to avoid, especially if rare classes are important.

According to Sun et al. [2009], the problem of imbalanced classes is especially salient for small datasets. The authors refer to research presented in a paper by Japkowicz and Stephen [2002], which shows that as the size of the training data increases, the error rate caused by the imbalanced class distribution decreases. So, imbalanced classes cease to be a problem if enough training data is given. However, how much data is enough is another question.

The imbalance in class distribution often comes together with overlapping classes. They make a system especially sensitive to class imbalance and significantly decrease its performance [Japkowicz and Stephen, 2002; Prati et al., 2004].

The problem of imbalanced data is particularly relevant to us. All the datasets we are going to work with are highly imbalanced - their instances are distributed between hundreds of classes, and many of these classes have only one or two inhabitants. In addition, the TRADR dataset is rather small. Ambiguity is another problem, e.g., up to 35% of instances in FrameNet and 53% in TRADR contain ambiguous lexical units. We will focus on these characteristics of our data in Chapter 3, and in this section we would like to introduce the most common methods for handling class imbalanced data. They can be roughly divided into two large groups: data level approaches and algorithmic level approaches [Alam et al., 2019]. It is, of course, possible to use a combination of techniques from both groups.

2.4.1 Data level approaches

To data level approaches belong techniques that modify the data in order to make it more balanced and decrease the effect of a majority class or classes. Normally, it is done before training. There are three main groups.

The **oversampling** approach involves increasing the number of instances of a minority class. The easiest way to do this is to replicate already existing instances. This can be done randomly or in an informed way, when the samples to copy are carefully chosen. A disadvantage of this method is a high possibility of a model overfitting, as the role of certain separate samples grows, but no new samples are created.

To avoid the overfitting problem it is possible to use techniques that assume replicating and creating new samples of a minority class in user-defined proportions. These techniques differ in their nature and complexity. Some of them generate new instances based on distance between pairs of neighbours in the minority class, e.g., SMOTE [Chawla et al., 2002] and its variants, others employ clustering to decide how many new instances need to be generated and to filter out irrelevant data, e.g., CBOS [Chen and He, 2014] or TRIM [Puntumapon et al., 2016]. There also exist approaches to synthesize new samples by adding some noise to already existing ones. Many algorithms

combine several methods. Additional information about them, as well as more detailed taxonomies can be found, e.g., in papers by Esteves [2020], Branco et al. [2015] or Sun et al. [2009].

The approaches described above are easy to use, and usually they perform better than a random oversampling. However, one should be careful using these techniques, as many of them synthesize new instances disregarding the majority classes. So, it may happen that the newly created samples lie too close to the representatives of a majority class, and this may lead to an overlapping of classes and decrease the performance of the model [Sevastianov and Shchetininb, 2020].

Undersampling is a technique that removes instances from a majority class to make the dataset more balanced. It also can be random or informed, when the algorithm tries to identify and eliminate noisy samples or samples that come from overlapping regions. The latter category includes, for instance, a cluster-based algorithm DSUS [Ng et al., 2014], and distance-based algorithms called RBU [Koziarski, 2020] and OSS [Kubat et al., 1997].

The main disadvantage of the undersampling approach is a high possibility of deleting some important data. As a result the classifier loses its ability to generalize [Sevastianov and Shchetininb, 2020].

Also, there is a sub-type of undersampling called active learning. It suggests that a classifier learns only from the most informative samples in the data. Finding these samples is not a trivial task. One of the approaches assumes using an SVM classifier to dissect the data with hyperplanes. Instances that lie closer to the hyperplanes are considered to be more important than those that lie farther away [Branco et al., 2015].

Hybrid methods combine the over- and undersampling approaches. An overview of such algorithms can be found, e.g., in [Esteves, 2020]. Note that some researchers use the notion of hybrid methods with respect to the techniques that use data level approaches together with algorithmic level ones and/or classifier ensembling (e.g., Branco et al. [2015]).

Both over- and undersampling are designed to alter the original class distribution inherent in the data and make it balanced. However, according to Sun et al. [2009], a balanced class distribution does not necessarily provide optimal results. And an optimal distribution for each particular case is unknown, so the right proportions of over- and undersampling are difficult to determine.

Effective resampling is another problem. Ideally, one would like to detect the sub-concepts in each minority class and then oversample each sub-concept. At the same time all irrelevant and noisy samples should be deleted from the majority classes. However, such data analysis can be very costly in terms of time and computation [Sun et al., 2009]. And the situation gets worse as the number of classes grows.

2.4.2 Algorithmic level approaches

This category includes various approaches that alter existing classifiers to make them more sensitive to a minority class. Using such techniques may help avoid performing resampling and save time and resources. On the other hand, any successful modification of a learning algorithm is only possible if one has a very good understanding of how it functions and why it fails working with imbalanced data [Sun et al., 2009].

One of the most well-known algorithmic techniques is **cost-sensitive learning**. It assumes different costs for different misclassification cases. These costs are usually given in the form of a matrix. The general idea is that it is possible to introduce higher costs (or penalties) for misclassification of rare class instances, and lower costs for misclassification of other instances. Thus, while trying to minimize the total cost, the classifier will focus more on the rare classes [Sun et al., 2009].

A drawback of cost-sensitive learning is that given a dataset, the corresponding cost matrix is usually unknown. In order to come up with such a matrix very good knowledge of the domain is necessary. And if the data has many classes, the problem gets even more complicated.

It is interesting to note that, according to Sun et al. [2009] and Branco et al. [2015], algorithmic level cost-sensitive learning cam be translated into a pure data level approach, which does not require a cost matrix. The trick assumes weighting the data space. It changes the original data distribution by multiplying each sample by a factor that is proportional to its importance. Again, the weights are usually unavailable, and the resulting model may be prone to overfitting [Branco et al., 2015].

Another subtype of algorithmic level approaches is formed by **ensemble-based techniques**. Note that some researchers place them into a separate category, as some techniques also incorporate data level methods (e.g., Sun et al. [2009] and Mahani and Ali [2019]).

Ensembling suggests training several classifiers and subsequently combining them into a single one. Classification is performed using a voting strategy - the predictions are aggregated and the label is chosen by the majority of votes [Mahani and Ali, 2019]. The motivation behind this approach is that the generalization ability of an ensemble is better than those of separate classifiers. This happens because each classifier is trained on certain data, so different classifiers most likely make different mistakes [Sun et al., 2009]. Ensemble-based approaches can be further divided into two subclasses: boosting and bagging.

Boosting incorporates cost-sensitive learning. At the very beginning of the training process equal weights are assigned to all instances, then a series of classifiers are trained one after another. After each classifier is trained the weights get updated so that misclassified instances cost more, and the next classifier is forced to focus more on such samples [Alam et al., 2019]. One of the most popular boosting algorithms is *AdaBoost*. For more information on *AdaBoost*, its variants, special modifications for imbalanced classes and other boosting techniques the reader can refer to the works by Sun et al. [2009], Esteves [2020] or Mahani and Ali [2019].

Bagging is a technique that assumes training several classifiers on subsets of a given dataset. The subsets are built using sampling with replacement. If the given dataset is imbalanced then either oversampling, or undersampling, or a combination of both is applied to create training sets [Mahani and Ali, 2019]. Bagging can be, of course, combined with boosting. Details on many modern bagging algorithms are given, e.g., in the works by Mahani and Ali [2019], Esteves [2020] or Alam et al. [2019].

For our experiments we will try out both data and algorithmic level approaches to handle imbalanced classes in our data. First, we will perform data oversampling, but instead of replicating or generating artificial TRADR examples, we will sample additional instances for rare classes from out-of-domain FrameNet and SALSA data. Second,

following Tan and Na [2019], we will consider integrating a frame filtering mechanism into our model. It assigns zero weight to the unlikely frame candidates based on the mapping of the given lexical unit to a set of frames it evokes. The mappings can be constructed before training using the training data or some external reference, e.g., FrameNet database. We will not introduce any changes to the loss function. Our approach can be especially helpful for non-ambiguous and/or rare frames.

2.5 Performance measures for imbalanced data

In Section 2.3 we showed how neural networks can be used to tackle multiclass problems. However, it is known that neural networks are sensitive to class imbalanced data [Sun et al., 2009]. As shown in Section 2.4, there exist many ways to handle class imbalance, but there is no guarantee that making the data balanced will help to train a more reliable classifier without hindering its ability to generalize. In addition to that, it is not always easy or possible to adapt the learning algorithm to handle imbalanced data.

This means that sometimes we cannot manipulate our data and/or classifier much, and need robust techniques that would allow to estimate the classifier performance in an adequate way. In what follows we will discuss some of the most widely used performance measures for a classifier trained on imbalanced data. In all cases, if not stated otherwise, we assume that a classifier produces discrete output. We will also explain why some common metrics, such as accuracy, are not reliable in this scenario.

In order to explain how many performance measures work, we will need a notion of a confusion matrix (or a contingency table). Table 2.3 shows an example of a binary confusion matrix. In this case we have two classes, one is called positive, i.e. it is a class we are interested in, and the other is called negative. Rows represent the true distribution of classes, and columns - the predicted one. Inner cells show the number of correctly (green) and incorrectly (pink) classified instances. The names of the cells are self-explanatory. False positives are usually called Type I error, and false negatives - Type II error. A binary confusion matrix can easily be extended to three and more classes. In this case instead of true/false positives and negatives we will simply have correctly and incorrectly classified samples.

		Predicted class	
		Negative	Positive
True class	Negative	true negatives (TN)	false positives (FP)
	Positive	false negatives (FN)	true positives (TP)

Table 2.3: Confusion matrix: example (taken from Tharwat [2020])

According to Tharwat [2020], any performance metric that uses both rows of the confusion matrix is sensitive to imbalanced data, and it cannot distinguish between the numbers of correct labels from different classes unless changes in class distribution cancel each other.

To explain this phenomenon, the author compares two metrics: accuracy and geometric mean. Accuracy is defined as $Acc = \frac{TN+TP}{TN+FP+FN+TP}$ and geometric mean as $GM = \sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}}$. Let us assume that the number of true negative samples changes by some value α. Now we have $Acc = \frac{\alpha TN+TP}{\alpha TN+\alpha FP+FN+TP} \neq \frac{TN+TP}{TN+FP+FN+TP}$, and $GM = \sqrt{\frac{TP}{TP+FN} \cdot \frac{\alpha TN}{\alpha TN+\alpha FP}} = \sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}}$. We see that both metrics use both rows of the confusion matrix for a score calculation, but while accuracy is affected by changes in class distribution, geometric mean is not, as α can easily be cancelled [Tharwat, 2020].

It is important to remember that accuracy may be the same for balanced and imbalanced data in case TN and TP stay the same. This may be misleading. And it is one more reason why accuracy should be avoided when dealing with imbalanced classes [Tharwat, 2020].

Among commonly used metrics that are also considered to be unreliable because of the reason presented above are precision, F-score (the harmonic mean between precision and recall), Matthews correlation coefficient (MCC), optimized precision (OP), Jaccard measure and some others [Tharwat, 2020]. We will not present them here. In what follows we will focus on metrics that will be directly used for the evaluation of our models, and on those that are necessary to understand how our primary metrics work.

2.5.1 Sensitivity and specificity

Sensitivity, which is also called true positive rate (TPR) or recall is a metric that shows the proportion of correctly classified samples with respect to all true positives. Specificity, or true negative rate (TNR) does exactly the same but for negative instances. The formulas for both sensitivity and specificity are given in Equation 2.1.

$$TPR = \frac{TP}{TP+FN} = \frac{TP}{P}, \quad TNR = \frac{TN}{TN+FP} = \frac{TN}{N} \tag{2.1}$$

Each of the metrics relies only on values from a certain row, so they can be considered safe when handling imbalanced data. However, there is a disadvantage. Sensitivity and specificity estimate different aspects of a classifier performance and cannot be used to assess the quality of a classifier in general [Tharwat, 2020].

2.5.2 False positive and false negative rates

Sensitivity and specificity have complement metrics - false negative rate (FNR) and false positive rate (FPR), respectively. So, FNR is a ratio of incorrectly classified positive samples with respect to all positive examples, and FPR is exactly the same for true negative samples. Like sensitivity and specificity, both metrics can be used with imbalanced data and assess separate aspects of classifier performance. Their formulas are presented in Equation 2.2.

$$\begin{aligned} FNR &= 1 - TPR = \frac{FN}{TP+FN} = \frac{FN}{P} \\ FPR &= 1 - TNR = \frac{FP}{TN+FP} = \frac{FP}{N} \end{aligned} \tag{2.2}$$

2.5.3 Geometric mean

Geometric mean (GM) is another performance metric suitable for imbalanced data. Originally, GM is defined as the nth root of the product of n numbers, and represents the so-called central tendency, i.e. a central or typical value for a probability distribution [Wikipedia, 2020c]. In machine learning GM is a square root of sensitivity and specificity product as depicted in Equation 2.3.

$$GM = \sqrt{TPR \cdot TNR} \tag{2.3}$$

One of the advantages of GM is its ability to alleviate the influence of outliers. This can be useful, for instance, in cases when sensitivity of a classifier is good, but the specificity is bad, or vice versa.

2.5.4 Index of balanced accuracy

One more useful metric that considers sensitivity, specificity and GM is called an index of balanced accuracy (IBA). As García et al. [2009] point out, many metrics for imbalanced data do not reflect the contribution of each class to the overall performance, and do not take into consideration which class is prevalent. So, e.g., different combinations of TPR and TNR may produce the same GM scores. To deal with this shortcoming, García et al. [2009] introduce an IBA metric shown in Equation 2.4.

$$IBA = (1 + \alpha \cdot Dominance) \cdot GM^2 = (1 + \alpha(TPR - TNR)) \cdot TPR \cdot TNR \tag{2.4}$$

According to García et al. [2009], the $Dominance = \text{TPR} - \text{TNR}$ metric is used to inform about the dominant class and how significant this dominance relationship is. The *Dominance* score lies in the range $[-1; 1]$. The more the value is close to zero, the more balanced both rates are. The IBA metric can be interpreted as a trade-off between *Dominance* and GM^2. Its score may vary between zero and one inclusively. To weight the value of *Dominance* a parameter $0 \leq \alpha \leq 1$ is introduced. If $\alpha = 0$, IBA turns into GM^2.

Table 2.4 compares TPR, TNR, accuracy and GM metrics with the IBA metric with three various α values. Assume that all the classifiers were trained on the same imbalanced dataset, which contains two classes in proportion 1:10 with a minority class being a positive one. One can see that the accuracy does not care about how well a classifier can recognize the positive class, and depends on the majority class rate. GM outputs almost the same score regardless of the performance for the positive class. And IBA is more flexible. If we want to give more weight to *Dominance*, the score will be biased towards the last classifier, which demonstrates the best performance for the positive class. The smaller the parameter α is, the more weight is laid on the performance with respect to the majority (negative) class.

Other useful metrics suitable for the evaluation of models trained on imbalanced data include diagnostic odds ratio (DOR), Youden's index (YI), discriminant power (DP), balanced classification rate (BCR), etc. More information about them can be found, e.g., in the surveys by Tharwat [2020] or Sokolova et al. [2006].

	TPR	TNR	$Acc.$	GM	IBA_1	$IBA_{0.5}$	$IBA_{0.1}$
Classifier 1	0.550	0.950	0.914	0.723	0.314	0.418	0.502
Classifier 2	0.680	0.810	0.798	0.742	0.479	0.515	0.544
Classifier 3	0.810	0.680	0.692	0.742	0.622	0.587	0.558
Classifier 4	0.950	0.550	0.586	0.723	0.732	0.627	0.543

Table 2.4: Comparison of several performance metrics and IBA with different α parameters (taken from [García et al., 2009])

In our work we will use index of balanced accuracy (IBA) as the main metric, because it has several advantages in comparison with other metrics that we mentioned. First, it evaluates all aspects of the model, i.e. its ability to recognize both positive and negative instances. Second, in contrast to some other metrics, e.g., DOR or DP, its score lies between zero and one, is better interpretable and can be compared with standard accuracy. Third, while some metrics like YI, GM or BCR do not always react correctly on changes in true positive and true negative rates (e.g., their scores stay the same if the rates are swapped), IBA does reflect such changes. Forth, IBA is always defined (e.g., DOR and DP are not) and it allows the user to control the influence of dominant classes with the α parameter. Finally, IBA (as well as some other metrics) was implemented as part of the *Python imbalanced-learn* package (see Lemaître et al. [2017]) and can be used out of the box.

2.6 Micro- and macro-averaging

If we have only two classes, the application of the performance metrics presented above is straightforward. However, if there are more than two labels, we need to calculate the scores for each class and then assess the overall performance of the classifier. This can be achieved using two different approaches: micro- and macro-averaging [Sokolova and Lapalme, 2009].

Micro-averaging assumes that only one confusion matrix is created. The counts, namely, true negatives, false positives, etc., are calculated for each separate class based on the 'one against all' method, i.e. one class is treated as positive, and the rest form the negative class. The counts for each class are accumulated in the same confusion matrix, using which various performance measures can be computed.

In case of macro-averaging, the counts for each class are gathered in separate matrices, using the same 'one against all' approach. All necessary scores are calculated based on each individual matrix and then averaged.

The difference between micro- and macro-averaging lies in their treatment of separate classes and samples. So, micro-averaging gives all samples the same weight, no matter if they come from the majority or minority classes. If most misclassified instances belong to the minority classes, this won't influence much the final score, as all samples are treated as equally important, and instances of large classes absolutely outnumber those of rare ones. Macro-averaging gives all classes equal weight. In the result the minority classes influence the final score as much as the majority ones. So,

if one is interested in the classifier performance on small classes, macro-average is an approach to go with [Van Asch, 2013].

		Predicted	
		N	P
True	N	10	0
	P	0	90

(a) Class A

		Predicted	
		N	P
True	N	95	0
	P	5	0

(b) Class B

		Predicted	
		N	P
True	N	95	0
	P	5	0

(c) Class C

		Predicted	
		N	P
True	N	200	0
	P	10	90

(d) Aggregated

Figure 2.7: Micro- and macro-averaging: confusion matrices

Let us illustrate both approaches with the following example. Assume that we have a classifier trained to distinguish between three classes A, B, C, and our test data contains 100 samples, of which 90 belong to class A, 5 to class B and another 5 to class C. Figure 2.7 demonstrates classification results in the form of four confusion matrices: matrices (a), (b) and (c) show the counts for each separate class, matrix (d) presents aggregated counts, i.e. the summed up counts from the first three matrices. Judging by the matrices, our classifier has perfect knowledge of the majority class, namely, class A, but fails to classify correctly both minority classes B and C. Assume that we want to evaluate our classifier using the sensitivity measure. Following the micro-averaging approach, we take matrix (d) and get $TPR = \frac{TP}{TP+FN} = \frac{90}{90+10} = 0.9$. According to the macro-averaging approach, we calculate averaged sensitivity, using the other three matrices: $TPR = (\frac{90}{90+0} + \frac{0}{0+5} + \frac{0}{0+5})/3 \approx 0.3$. It is clear that two scores are very different, so if the correct recognition of minor classes is important for the task, it is recommended to use macro-averaging for evaluation of a classifier performance [Van Asch, 2013].

As in our research we are interested in recognition of all the classes regardless of the number of inhabitants, the evaluation of all the models we are going to train will be done with macro-averaging. The package *imbalanced-learn* that we will use applies it by default for all the metrics.

So, in this chapter we have provided the reader with the most essential knowledge related to our research questions. We have presented the theory of frame semantics and the FrameNet project, defined the most important concepts and illustrated how the meaning of a sentence can be captured using the frame structure. We have also discussed neural networks as one of the most widely used approaches to multiclass classification, and examined in detail the components of the BERT model. We presented various techniques to handle imbalanced data and introduced performance measures suitable for models trained on such data. We have tried to explain why this information is relevant for our work, and what approaches we are going to use exactly and how.

Chapter 3

Data

In this chapter we will familiarize the reader with the TRADR, FrameNet and SALSA data. We will compare the corpora with respect to their domains, the distributions of semantic frames, lexical units and their parts of speech. We will also examine the role of ambiguous lexical units in our data. Such information will help understand the nature of the data, predict the results we can expect from the frame classifier, motivate our choice of performance metrics and provide some insight into why the classifier may make certain mistakes. Besides general data analysis, we will also explain how we prepared and split all the data into training, validation and test sets.

3.1 TRADR data

TRADR is a research project in the area of robot-assisted disaster response [Kruijff-Korbayová et al., 2015; Disaster Robotics Research Project, 2020]. As part of the project several teams of firefighters performed a series of exercises that simulated situations after a disaster, such as a fire, an explosion and so on. Team communication recorded during these exercises was later transcribed and is known now as the TRADR corpus. It consists of 15 files with dialogues, six files contain dialogues in English, and nine - in German. Six German dialogues were translated into English in the course of the TRADR project in order to get more English training data.

In what follows we will examine the data in detail. It is important to mention that the analysis of the TRADR domain, characteristics of team communication and dialogue contents that will be given below is relevant for both English and German sub-corpora. The rest of the current section will be devoted to the investigation of the distributions of semantic frames, lexical units and their parts of speech in English TRADR data (both original and translated). The corresponding analysis of the German sub-corpus will be presented separately in Section 3.4, due to the differences in annotation approaches.

3.1.1 Domain

So, all TRADR dialogues represent human-human team communication in robot-assisted disaster response. The dialogues in German were recorded in 2015 and 2016, the English data comes from 2017. It should also be pointed out that the firefighters who

took part in the exercises in 2017 were not native English speakers - they were asked to use English for the experiments. In total the joint corpus contains about 2,9k dialogue turns (see Table 3.1).

Recording	Mission	Duration	Turns
TJex2015			**363**
	Day 1	48:21 min	186
	Day 2	33:21 min	177
TEval 2015			**1,279**
	Day 1	58:23 min	359
	Day 2	65:04 min	356
	Day 3	57:15 min	272
	Day 4	53:22 min	292
TEval 2016			**422**
	Day 1	n.a.	312
	Day 2	n.a.	110
TEval 2017			**811**
	Day 1	64:02 min	239
	Day 2	149:20 min	400
	Day 3	56:36 min	172

Table 3.1: TRADR data: corpus composition (based on the paper by Anikina and Kruijff-Korbayová [2019])

A typical field exercise assumes that a rescue team explores the environment using robots, namely unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs). The robots look for sources of smoke, fire, or contamination, check if there are victims and if the site is safe enough for human rescuers and firefighters to enter. The robots are equipped with gas detectors and two camera types: a standard camera and an infrared one, they can take pictures and share them. Some UGVs have a mechanical arm and are capable of taking samples, as well as turning, pushing or moving objects. They can also climb stairs. Some pictures of UGVs taken during TRADR exercises can be seen in Table 3.2.

The team consists of several operators who control the robots, a team leader (TL) and sometimes also a mission commander (MC). The MC is in charge of the whole mission and gives tasks to teams. The TL distributes the tasks between the operators, coordinates their actions and reports to the MC. The operators use robots to perform the tasks assigned to them and report to the TL about the results or possible difficulties. The robots are controlled remotely, and the operators and the TL do not have to be present on site directly. Instead, they can see the area and all the objects there with the help of robots' cameras. The robots investigate the unknown and continuously changing environment step by step from different angles. In parallel a virtual map with relevant points of interest marked on it is being created.

3.1.2 Characteristics of team communication

Now we would like to present the main characteristics of team communication in the above given scenario and illustrate them with some examples. This will give the reader

an insight about how exactly the communication between mission participants is organized.

First, all information flows through a rather **complex communication pipeline** with several participants. This leads to situations when the same information or requests are repeated twice. Example 3.1.1 shows how a question about a leakage is transmitted from the MC to the operator via the TL. In a similar manner the answer is transferred back to the MC.

Example 3.1.1. *Information flow from MC to Operator via TL*
MC: I got the picture of the blue canister. According to the result of my request it seems to be fuel. The question is if the canisters broken, or has a leakage, or if it's still tight?
TL: I'll check that.
TL: UGV-2 to team leader.
TL: UGV-2, please answer.
TL: Yes, the canister with the Otalin, is it broken? Is there only a small puddle or has it leaked out completely? How many litres are that?
UGV-2: Well, I can't exactly say how much, but there's definitely some leaking.

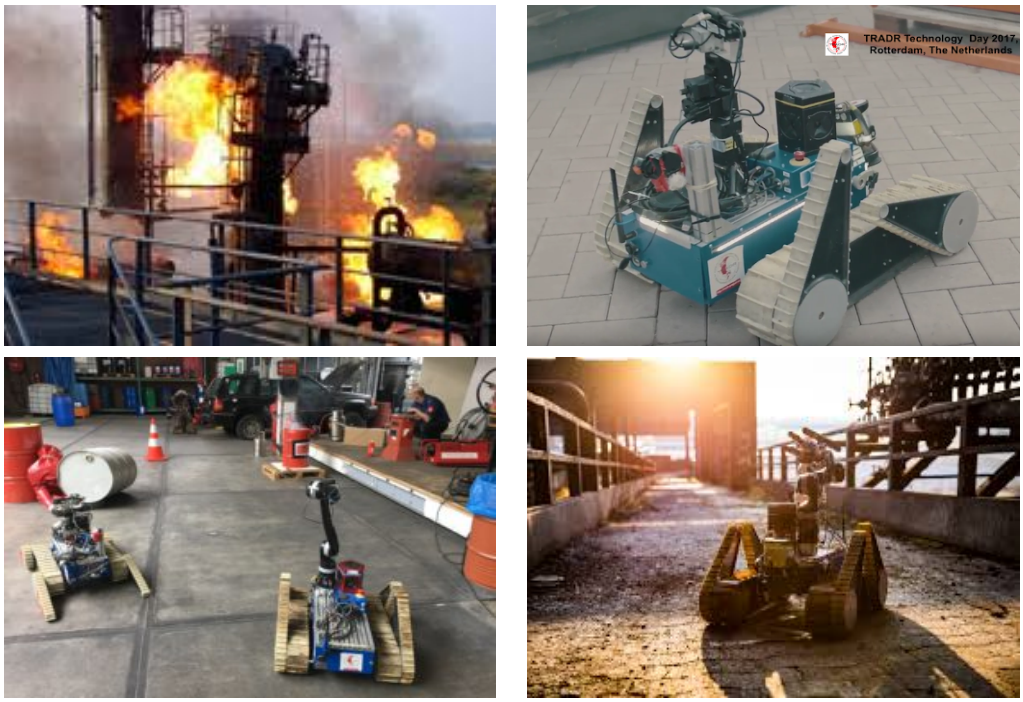


Table 3.2: UGVs in action (taken from [Disaster Robotics Research Project, 2020])

Second, participants are supposed to follow a certain **communication protocol**, i.e. there are special phrases to be used to start/finish conversation, check the connection quality, accept/reject requests, etc. Communication protocol helps to make the communication better structured and more clear, especially, if communication is done by radio. Example 3.1.2 presents a typical conversation opening, and illustrates phrases used to indicate that a dialogue turn is finished (*'over'*) or successfully processed (*'roger'*).

Example 3.1.2. *Following communication protocol*
UGV-2: Team leader to UGV 2 please answer.
TL: I'm listening.
UGV-2: Yes, I've taken two pictures again, one of the smoke development and one of the green barrel that's lying here. Over.
TL: Yes, roger.

Third, TL constantly switches from one operator to another, and the flow of information is usually split into **several threads** that may overlap. Sometimes, this may lead to confusion and misunderstanding. One of such situations is depicted in Example 3.1.3, where operators accidentally swapped their tasks. Sometimes the TL speaks with two operators simultaneously, as Example 3.1.4 demonstrates.

Example 3.1.3. *Accidental task swap*
TL: Delta, how are you doing? Over.
UGV-2: I'm looking for the victim one that... You've told me inspect victim one.
TL: Delta, you said you're speaking for the victim. Is that correct?
UGV-2: Yeah, that's correct. That's my screen.
TL: Erm... I thought, you should go to the fire. And look how big the fire is, and how the heating is in the surrounding. Over.

Example 3.1.4. *TL speaking with two operators at the same time*
TL: UGV-2 or UGV-1, do you have gas alert in your UGV? Over.
UGV-1: UGV-1. Negative.
UGV-2: UGV-2. No alert. Over.
TL: All right.

Next, because the mission participants can observe the environment via a virtual map or pictures made by robots, they often **refer to objects on the screen using gestures or pointing**. Example 3.1.5 illustrates the situation.

Example 3.1.5. *Reference to location via pointing*
UGV-2: Where is that?
TL: There. That is on this corner. Over here.
UGV-2: Ok. Ok.
TL: Emm... Over here.
UGV-2: Ok.
TL: Now you go explore this corner for more problems.

The fact that the mission participants perceive the environment via some medium is reflected in language usage. We call this **double reality representation**. Often, when the TL assigns tasks and gives commands, they speak to an operator, but mean a robot. The operators may mark important objects and places that the robots investigate on the virtual map as *'points of interest'* (POIs). These POIs are then used to refer to the real objects or places, although they are just symbols on the computer screen. Both phenomena are shown in Example 3.1.6, cases *(a)* and *(b)* respectively.

Moreover, an unexpected expression referring to some area or object may pop up during the dialogue simply because this area or object was 'silently' put on the screen, and this fact was never explicitly mentioned before.

Example 3.1.6. *Double environment representation*
(a) TL: UGV-2, UGV-1. Can you go forward a little bit?
UGV2: I will go forward. Operator one, so I make your way clear. Over.
(b) TL: Can you go to a victim forty and check for functions? Over.

Finally, like any **spontaneous speech** TRADR dialogues are characterized by repetitions, elliptical constructions and a heavy use of fillers, such as 'erm', 'uh', etc. Some utterances are not finished, and often they contain grammatical mistakes.

So, above we have summarized the main features of the TRADR team communication. We have seen that some of them, e.g., communication by protocol, complex communication pipeline or double reality representation are domain-specific. They give us a better understanding of the domain and help explain some phenomena in frame distribution, e.g., why some frames are much more common than others.

3.1.3 Domain-typical topics

To give the reader some idea about the actual contents of the TRADR dialogues, in this section we will briefly summarize the topics that are the most characteristic of our data. We will also mention the most common problems that the mission participants had to deal with during the exercises, as the discussion of these problems makes a significant part of the dialogues. Later we will see how the topics presented here correlate with the distribution of semantic frames.

Let us start with dialogue contents related to **mission tasks**. As mentioned earlier, during a mission robot operators get requests from the team leader and report back the results. Taken together, requests, progress reports and questions/answers related to them form the bulk of the dialogues. In general we can distinguish the following requests: explore certain area and find sources of smoke, fire, contamination, etc.; search for victims; move or fly along a certain path; mark mission-related objects and/or way-points on a virtual map; make pictures and share them. To this group we can also add all sorts of questions concerning task understanding, acceptance or confirmation.

Quite a lot of utterances serve the purpose of **establishing or checking connection** between the mission participants. Obviously, the most common problem here is a poor connection, so that the participants mishear or misunderstand each other (see Examples 3.1.7 and 3.1.8).

Example 3.1.7. *Poor connection: radio's down*
MC: Team leader to mission commander. Please answer.
UGV-1: Team leader, radio's down.
MC: Okay.
TL: Alright. The team... The team leader can hear you again. Radio's working again.

Example 3.1.8. *Poor connection: misunderstanding*
TL: Can you explore that area, please?
UGV-2: Yeah. What's area do you mean?
TL: Area on the west side. Over.
UGV-2: The area on the left side. Ok.

TL: The area on the west side. Over.
UGV-2: On the west side.

Another dialogue topic in a constantly changing environment is **tracking changes**. It can be changes in robots' locations and status, or changes of the surroundings, e.g., smoke and fire may spread, victims may change their posture or move (see Examples 3.1.9 and 3.1.10).

Example 3.1.9. *Tracking changes: robot's status*
TL: Can you give me your status? Because I don't see you are really moving.
UGV-2: I've seen that to myself too. One moment. I... I will give my status for about thirty seconds. Over.

Example 3.1.10. *Tracking changes: environment*
UGV-1: One victim is outside the plant. Is going away. And victim two, I guess he is dead.

Discussion of various **technical problems** also happens quite often in the TRADR dialogues. It can be problems with internet connection, with sending pictures and data, receiving GPS signal or problems related to controlling the robots or drones, including cases with empty batteries, troubles with joystick, flippers or arm (see Examples 3.1.11 and 3.1.12).

Example 3.1.11. *Technical problems: sharing pictures*
TL: Ground operator one, the picture hasn't arrived yet.
UGV-1: Team leader, do you have a picture now?
TL: Negative.
UGV-1: Yes, it doesn't work somehow.

Example 3.1.12. *Technical problems: robot stopped working*
UGV-1: Team leader. My robot has stopped working.
TL: Both robots stopped. Can you reset them? Over.
UGV-1: Negative. For operator one.
UGV-2: Negative for two.

Another common problem during field exercises has to do with **reality interpretation**. The problem is usually caused by indirect perception of the environment by mission participants. As the team members see the site mostly through robots' cameras, and there can be smoke, gas or fire, it is not always easy to identify an object in sight, or tell for sure that the robot has found what it was looking for. Sometimes the operators report about the same thing thinking that they have found two separate ones. The speakers may be unsure if they discuss the same entity or different ones. These problems are illustrated by Examples 3.1.13 and 3.1.14.

Example 3.1.13. *Difficulties in distinguishing two objects*
UGV-1: I see a victim. It's looks like he's sitting on a chair. Is that the same victim you see?
UGV-2: Negative. It's an... erm... maybe. My victim is also sitting on an chair.

Example 3.1.14. *Identification problem*
MC: On UAVs first picture showing the overview there is a light spot. Maybe you can zoom in again? I don't know what that is.

Finally, quite frequent among English TRADR dialogues are cases when the dialogue participants try to **establish a correspondence** between points of interest on a virtual map (screen) and real objects or places. Also, the situation is sometimes complicated by the fact that environment may change. Especially frequent are problems with identifying the robots' location. Some of such difficulties are shown in Examples 3.1.15 and 3.1.16.

Example 3.1.15. *Difficulties in identifying robot's location*
TL: Yeah okay, I should be able to see you then. It's in front of the smoke machine, right?
UGV-1: Nope. I'm in front of that furnace that has various portholes. Well, that's how I would describe it.

Example 3.1.16. *Difficulties in matching POIs with real objects*
UGV-1: Victim eleven and smoke five. I put POI.
TL: So, victim eleven is that victim which is code three and smoke five, smoke fifteen... Is that the barrel?
UGV-1: Yes. Smoke five is the barrel. Right.
TL: Eeeh. Fifteen, isn't it?

To conclude our survey of domain-typical topics, we would like to point out that while the discussion of mission tasks, their progress and status is in the focus of team communication, a considerate part of the dialogues is devoted to the discussion of various problems that may accompany the mission, e.g., bad radio/internet connection, difficulties in establishing correspondence between real objects and symbols on the map, tracking changes and so on. In Section 3.1.6 we will see how the topics that we have discussed found their reflection in the distribution of semantic frames.

3.1.4 Dialogue turns distribution

Let us first present some general statistical facts about the TRADR data. This will help better understand the peculiarities of team communication in the disaster response scenario. In order to process the dialogues, split them into separate sentences and perform tokenization we use the NLTK library for *Python* [Bird et al., 2009].

Table 3.3 shows the distribution of dialogue turns, utterances and tokens between all mission participants in both English and German dialogues. Also, average numbers of utterances per turn and tokens per utterance are given. The upper part of the table demonstrates information about the original German dialogues, next is the statistics of all the available translations from German into English. Figures that characterize the original English dialogues follow. The last part of the table presents the statistics of the joint data, which combines original English dialogues and six German dialogues that were translated into English.

We see that in total the data in English contains 2,094 dialogue turns, and the number of turns in dialogues translated from German is about 1.6 times bigger than

the number of turns in the English ones. On average, each dialogue turn contains 1.23 utterances, and English turns are slightly longer than the German ones. All turns taken together contain 22,156 tokens, and a single utterance is rather short - on average it consists of 8.68 tokens. One can see that as a rule a single utterance translated from German into English is about 1.7 times longer than a single utterance recorded in English - on average they have 10.79 and 6.48 tokens respectively.

	MC	**TL**	**UGV-1**	**UGV-2**	**UAV**	**Total**
German data						
# Dialogue turns	60	984	466	250	304	2,064
# Utterances	61	997	470	250	307	2,085
# Tokens	526	6,165	3,630	2,002	2,243	14,566
Avg. # utterances per DT	1.02	1.01	1.01	1.00	1.01	1.01
Avg. # tokens per utterance	8.62	6.18	7.72	8.01	7.31	6.99
English data translated from German (6 dialogues out of 9)						
# Dialogue turns	60	586	286	168	183	1,283
# Utterances	61	596	289	168	186	1,300
# Tokens	820	5,214	3,620	2,131	2,246	14,031
Avg. # utterances per DT	1.02	1.02	1.01	1.00	1.02	1.01
Avg. # tokens per utterance	13.44	8.75	12.53	12.68	12.08	10.79
Original English data						
# Dialogue turns	-	427	224	157	3	811
# Utterances	-	710	309	231	3	1,253
# Tokens	-	4,769	1,872	1,461	23	8,125
Avg. # utterances per DT	-	1.66	1.38	1.47	1.00	1.55
Avg. # tokens per utterance	-	6.72	6.06	6.32	7.67	6.48
All data (original English data + English data translated from German)						
# Dialogue turns	60	1,013	510	325	186	2,094
# Utterances	61	1,306	598	399	189	2,553
# Tokens	820	9,983	5,492	3,592	2,269	22,156
Avg. # utterances per DT	1.02	1.29	1.17	1.23	1.02	1.23
Avg. # tokens per utterance	13.44	7.64	9.18	9.00	12.01	8.68

Table 3.3: TRADR data: dialogue turns (DT) statistics

It is interesting to compare the English translations of the German dialogues with their original versions. A noticeable difference can be seen in the total number of tokens used, and as a consequence, in an average utterance length. So, in total, six translated dialogues contain almost as many tokens as the nine original dialogues with sentences being approximately 1.5 times longer than the original ones.

Looking at Table 3.3 we can also notice that the team leader has more turns than each of the other mission participants: in total this number is only slightly smaller than the sum of all the rest dialogue turns taken together. The least number of turns belongs to the mission commander, who is actually present in only two of all TRADR dialogues in German, and in field exercises performed in English in 2017 this role is absent. Next, we see that UAV plays much more notable role in exercises done in German: 304 turns

against 3 turns in the English dialogues. As for average utterance length, the longest ones belong to mission commander, and the utterances of the team leader are usually shorter than those of robot operators.

So, the main takeaways for us here are as follows. First, our English and German TRADR corpora are quite small (22,156 and 14,566 tokens respectively). Second, the dialogue turns of all mission participants are really short - their length hardly exceeds one utterance. Third, the utterances themselves are also short - normally, not longer than seven or nine tokens.

3.1.5 Utterance completeness

As normally we use semantically complete sentences containing frame-evoking elements for training/testing a model, it is also important to analyse the completeness of utterances contained in the TRADR corpus.

First, it is necessary to point out that, due to the fact that we deal with spontaneous speech, the dialogues contain many communication fragments, which represent all sorts of noises, unfinished sentences, isolated words, etc. As they are incomplete from the point of view of semantics, we simply assign a frame *'Communication_fragment'* to all of them. Such fragments do not have frame-evoking targets, so they are excluded from training/testing process. In total, communication fragments make up about 2.5% of all the data. Because English was not the native language of the firefighters who took part in the field exercises in 2017, communication fragments are much more frequent in the English TRADR dialogues than in the German ones.

Second, the data also contains many elliptical utterances. We label an utterance as elliptical if we can deduce its meaning, but the target that evokes this meaning is actually missing. There may be different reasons why the target is missing, from cases where it is omitted to avoid redundancy to cases with grammatical mistakes or slips of the tongue. About 13.5% of all our data is elliptical. Because of the missing targets we have to exclude this part of data from the experiments as well.

Almost 84% of data are full utterances with targets, and we use them for training/testing procedures.

A summary of the distribution of utterances with different level of completeness across English and German dialogues can be found in Table 3.4. For each utterance type the table shows the number of its raw occurrences as well as the proportion of these occurrences in the corpus. The last row shows the total number of instances which have frame labels. Additional information about communication fragments and elliptical sentences, as well as examples, is given in Appendix A.

	Eng.		transl. from Ger.		All data	
Utterance type	# occ.	%	# occ.	%	# occ.	%
Communication_fragment	103	6.56	3	0.11	106	2.53
Elliptical utterances	295	18.78	270	10.31	565	13.48
Full utterances	1,173	74.67	2,347	89.58	3,520	83.99
Total	1571	100	2620	100	4191	100

Table 3.4: English TRADR data: utterances' completeness

Note that according to Table 3.4 the total number of utterances in all the dialogues is bigger than the total number of utterances given in Table 3.3: 4,191 vs. 3,338 sentences. This is due to the fact that our data was first annotated with dialogue acts, and in the process some utterances were split or merged or cleaned up, e.g., all fillers and hesitation markers within utterances were cut out and kept as separate dialogue acts. Frame annotation was built upon the dialogue act annotation with some sentences split/merged back into their original forms. Moreover, some utterances were assigned several frames, because several tokens within those utterances were interpreted as targets (see Appendix A for details).

Now let us take a closer look at the elliptical utterances. Currently they are excluded from our experiments, but can be relevant for future research. So, elliptical utterances do not have targets, but we are still able to assign frames to them. Table 3.5 shows top five most frequent frames that were assigned to the elliptical utterances, as well as the absolute and relative occurrences of these utterances in the corpus. The table is split into three parts. The first part focuses on the original English data, the second part - on the translations from German, and the last one - on all the available TRADR data in English. We see that almost 69% of the elliptical utterances belong to *'Communication_by_protocol'* frame. Other common frame labels here are *'Identity'*, *'Motion'* and *'Inspecting'*.

Eng.			transl. from Ger.			All data		
Frame label	# occ.	%	Frame label	# occ.	%	Frame label	# occ.	%
Comm._by_protocol	206	69.83	Comm._by_protocol	183	67.78	Comm._by_protocol	389	68.85
Motion	14	4.75	Identity	22	8.15	Identity	27	4.78
Inspecting	12	4.07	Holding_off_on	9	3.33	Motion	21	3.72
Communication	9	3.05	Motion	7	2.59	Holding_off_on	18	3.19
Holding_off_on	9	3.05	Create_representation	7	2.59	Inspecting	13	2.30

Table 3.5: English TRADR data: top 5 elliptical frames

Please, note that both Table 3.4 and Table 3.5 do not contain any information about the original German data. This is because elliptical constructions are treated differently there (see Section 3.4).

3.1.6 Semantic frames distribution

Before we start discussing the distribution of semantic frames in the English TRADR data, we need to know more about how the corpus was annotated. We annotated utterances in TRADR dialogues with frame-evoking targets, corresponding lexical units, frames and parent frames. We did not annotate them with frame elements, phrase types or grammatical functions. We assumed that each utterance can potentially have several targets. The detailed description of our approach to the annotation of TRADR data, as well as illustrating examples are presented in Appendix A. It also includes a discussion of such annotation issues as the absence of lexical units and senses in the FrameNet database, the choice of targets within the utterance, adaptation of frames and so on. Performing annotations, we had to introduce several new frames, their definitions can be found in Appendix B.

Now we can proceed with an analysis of the semantic frame distribution in the English TRADR data. This will help make some judgments about the semantic structure of the data and support our statements about the most common topics discussed earlier. In what follows we will take into consideration all frame labels present in our data regardless of the utterance type, i.e. communication fragments and elliptical sentences will also be a part of the analysis.

From Table 3.4 we know that the whole corpus contains 4,191 frame instances. These instances are distributed between 190 different frame labels. The original English data counts 107 different frames, and the part translated from German has 162. The distribution of the frame labels is not uniform, i.e. some frame labels occur much more often than others.

As we have too many frame labels to place them all into a single table, a diagram or a pie chart, we will focus only on the most frequent/infrequent frames. Table 3.6 demonstrates the proportions of such frames in the whole corpus. The left part of the table presents the top ten most frequent frames, their absolute and relative occurrences in the data, e.g., the frame 'Capability' is represented by 305 instances in the English TRADR corpus, and taken together these 305 instances make 8.35% of all the data. The right part of the table reveals the role of the infrequent frames, namely the number of individual frames and the amount of instances each of these frames has in the corpus, as well as the role of these instances in the data. As the corpus has lots of infrequent frames, we group them by the number of occurrences, e.g., 30 different frames in English TRADR data are represented by two instances each, so altogether we have 60 instances that make about 1.43% of all the data.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Communication_by_protocol	759	18.11	46	1	1.10
Communication_response_message	350	8.35	30	2	1.43
Capability	305	7.28	10	3	0.72
Motion	217	5.18	12	4	1.15
Perception_experience	217	5.18	15	5	1.79
Sending	140	3.34	6	6	0.86
Locative_relation	140	3.34	10	7	1.67
Create_representation	127	3.03	5	8	0.95
Identity	123	2.93	2	9	0.43
Communication_fragment	106	2.53	1	10	0.24
Total	2484	59.27	137	433	10.33

Table 3.6: English TRADR data: frame distribution in all dialogues

One can easily see that the utterances that belong to the top ten most frequent frames make almost 60% of the corpus, while the instances of 137 infrequent frames (each has maximum ten inhabitants) count only about 10%. The frame distribution is heavy-tailed: 137 infrequent labels make about 72% of the total number of different frames, and top ten most frequent ones - only about 5%. The plots illustrating the situation can be found in Table 3.1. They show the dependence of frame frequency on its rank, where the rank means a frequency's place in a list of frequencies sorted from

the largest to the smallest. In the left plot the axes have linear scales, in the right one both axes are logarithmic.

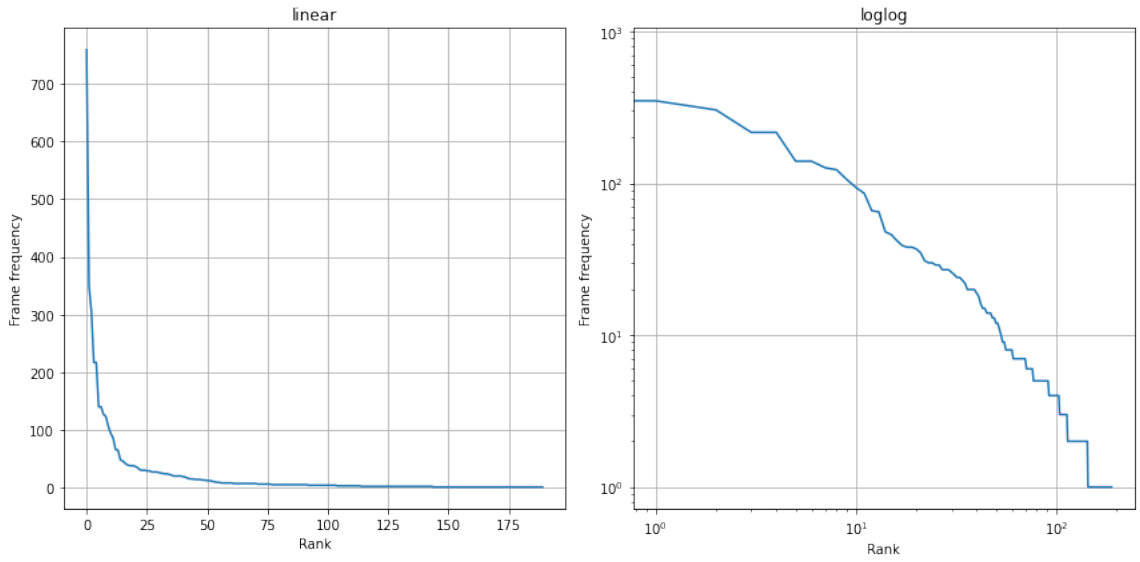


Figure 3.1: TRADR data: frame frequencies

If we take a look at the most frequent frames presented in Table 3.6, we will see that *'Communication_by_protocol'* is the most common frame, its instances form more than 18% of all the data. This is more than the sum of all occurrences of frames *'Communication_response_message'* and *'Capability'*, which are the second and the third most frequent frames respectively. Other widely used frames reflect the most common dialogue topics and mission tasks, e.g., *'Motion'*, *'Perception_experience'* or *'Create_representation'*.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Communication_by_protocol	248	15.79	29	1	1.85
Communication_response_message	123	7.83	18	2	2.29
Capability	110	7.00	12	3	2.29
Perception_experience	109	6.94	3	4	0.76
Communication_fragment	103	6.56	8	5	2.55
Motion	103	6.56	3	6	1.15
Inspecting	73	4.65	2	7	0.89
Locative_relation	64	4.07	3	8	1.53
Communication	52	3.31	1	9	0.57
Create_representation	50	3.18	4	11	2.80
Total	1035	65.88	83	262	16.68

Table 3.7: English TRADR data: frame distribution in the original dialogues in English

Similar tendencies can be observed in the English and German (translated) parts of the corpus, if we take them separately. The top three most frequent frames are identical

for both sub-corpora, other common frames in a slightly different order are also present in both sub-parts of TRADR data. The ratio of frequent frames to infrequent ones is similar as well. The details can be found in Tables 3.7 and 3.8.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Communication_by_protocol	511	19.50	51	1	1.95
Communication_response_message	227	8.73	19	2	1.45
Capability	195	7.44	15	3	1.72
Motion	114	4.35	9	4	1.37
Perception_experience	108	4.12	7	5	1.34
Sending	103	3.93	4	6	0.92
Identity	84	3.21	5	7	1.34
Create_representation	77	2.94	3	8	0.92
Locative_relation	76	2.90	4	9	1.37
Scrutiny	54	2.06	1	10	0.38
Total	1549	59.12	118	334	12.75

Table 3.8: English TRADR data: frame distribution in dialogues translated from German

Our analysis allows us to draw the following important conclusions. First, all English TRADR data is highly imbalanced - a few majority classes have hundreds of instances, while lots of minority ones are represented by only ten and less examples each. This fact motivates the choice of performance metrics for the evaluation of the frame classifier that were discussed in Chapter 2. Second, frames are good indicators of the dialogue contents presented in Section 3.1.3. E.g., the frequency of the *'Communication_by_protocol* frame shows that establishing connection really makes a large part of the dialogues, and *'Capability'*, *'Communication_response_massage'* and frames capturing various activities reflect discussion of mission relevant tasks, as well as the nature of these tasks.

3.1.7 LU and POS distributions

In what follows we will analyse the distributions of lexical units (LUs) and their parts of speech (POS) across the English TRADR corpus. This information together with the analysis of the ambiguity of LUs that will be discussed next is important for our experiments with the filtering of candidate frames by the given LU.

In total the English TRADR data counts 434 different LUs. Table 3.9 shows the top ten most frequent ones. All three parts of the table demonstrate similar tendencies: we see that the most frequent LUs are verbs, and they are mostly the same everywhere. The distribution of LUs, like the distribution of frames, is not uniform: the same top ten most common LUs occur in about 40% of all the utterances and at the same time make only slightly more than 2% of the whole amount of different LUs.

Looking at the lexical units in Table 3.9 it is easy to suggest the frames they may evoke, e.g., the LU *'see.v'* most likely evokes the frame *'Perception_experience'*, *'can.v'* - *'Capability'* and so on. However, the order of these frames would be a little

bit different from that presented in Table 3.6. Moreover, the most probable frame candidates for such LUs as *'listen.v'* and *'answer.v'* - frames *'Perception_active'* and *'Communication_response'* are absent from the list of top ten most frequent frames. This allows us to conclude that some LUs are ambiguous, i.e. they have several senses and thus evoke more than one frame. We will discuss the ambiguity of lexical units in Section 3.1.8.

Eng.			transl. from Ger.			All data		
LU	# occ.	%	LU	# occ.	%	LU	# occ.	%
can.v	106	9.04	can.v	192	8.18	can.v	298	8.47
see.v	76	6.48	listen.v	165	7.03	listen.v	168	4.77
go.v	73	6.22	roger.intj	150	6.39	roger.intj	161	4.57
be.v	36	3.07	answer.v	135	5.75	see.v	156	4.43
ok.intj	36	3.07	be.v	98	4.18	answer.v	135	3.84
okay.intj	32	2.73	send.v	89	3.79	be.v	134	3.81
explore.v	29	2.47	see.v	80	3.41	send.v	117	3.32
send.v	28	2.39	take a picture.v	54	2.30	go.v	113	3.21
thank.v	26	2.22	there be.v	50	2.13	there be.v	60	1.70
know.v	22	1.88	go.v	40	1.70	take a picture.v	59	1.68
Sum	464	39.56	Sum	1,053	44.87	Sum	1,401	39.80

Table 3.9: English TRADR data: top 10 most frequent LUs

Now let us briefly examine the POS distribution of lexical units presented in Table 3.10. Like the distributions of frames and LUs, it is also not uniform. All LUs are distributed between seven different POS tags. One can notice that within the whole corpus verbal LUs occur in 75% of the utterances. The second frequent POS that a LU may have is interjection. To this POS belong almost 8% of all the LUs. Interjections as frame-evoking elements are typical for *'Communication_response_message'*, which is quite common. It is also interesting that adjectival LUs are more frequent than nominal ones, and that prepositional LUs occur as often as the adverbial LUs.

Eng.			transl. from Ger.			All data		
POS	# occ.	%	POS	# occ.	%	POS	# occ.	%
.v	828	70.59	.v	1829	77.93	.v	2657	75.48
.intj	98	8.35	.intj	172	7.33	.intj	270	7.67
.n	98	8.35	.a	147	6.26	.a	190	5.40
.adv	53	4.52	.n	77	3.28	.n	175	4.94
.prep	52	4.43	.prep	63	2.68	.prep	115	3.27
.a	43	3.67	.adv	59	2.51	.adv	112	3.18
.conj	1	0.09	-	-	-	.conj	1	0.03
Total	1,173	100	Total	2,347	100	Total	3,520	100

Table 3.10: English TRADR data: LU POS distribution

3.1.8 Ambiguity of LUs

Ambiguity of lexical units is another important aspect we need to discuss, because this is one of the things that make training a frame classifier a challenging task.

Let us first check the top ten most ambiguous words in the TRADR corpus given in Table 3.11. As usual, the data in the table shows the results for the original English TRADR data, dialogues translated from German, as well as both parts taken together. The first column in each part contains LUs, the second one - number of different frames a certain LU evokes. One can see that all LUs in the table are rather common verbs, and they are mostly the same in all three parts of the table. According to the last part of the table representing the whole corpus, the LUs *'be.v'* and *'do.v'* are the most ambiguous - each of them evokes six different frames. There are four other LUs each of which evokes five separate frames. Note that three LUs in this column are also present among the top ten most frequent ones.

Eng.		transl. from Ger.		All data	
LU	# diff. fr.	LU	# diff. fr.	LU	# diff. fr.
do.v	5	there be.v	5	be.v	6
can.v	3	get.v	5	do.v	6
look.v	3	be.v	5	look.v	5
give.v	3	discover.v	4	there be.v	5
be.v	3	look.v	4	get.v	5
come.v	3	make.v	3	make.v	5
copy.v	2	escape.v	3	say.v	4
change.v	2	can.v	3	come.v	4
there be.v	2	come.v	3	can.v	4
check.v	2	take.v	3	discover.v	4

Table 3.11: English TRADR data: top 10 most ambiguous LUs

As lexical ambiguity represents a challenge for any frame classifier, we are naturally interested in how many lexical units in the TRADR data are ambiguous, and if these ambiguous lexical units are frequent. This information is summarized in Table 3.12. The first column of this table shows how many different LUs evoke a certain number of frames. The number of the evoked frames is given in the second column. So, e.g., in the whole corpus there are nine LUs, each of which evokes 3 different frames. The remaining two columns show the proportions of both ambiguous and non-ambiguous LUs with respect to all individual LUs taken together (the third column), and with respect to all their instantiations in all available senses (the fourth column). Using Table 3.12 we can observe that in general only about 15% of all LUs are ambiguous. However, all these ambiguous LUs are realized in nearly 53% of utterances containing targets. Simple calculations show that on average a single LU evokes 1.24 frames.

So, we see that while the ambiguous LUs are not very frequent in comparison to non-ambiguous ones, the frames that they evoke are frequent, and this may become a problem for the frame classifier, as it is not always possible to perform frame disambiguation using utterance context. The fact that we work with team communication data makes this problem even more complicated, as the mission participants often tend to omit many details to make their utterances brief and simple. We will return to the influence of ambiguity in Chapter 4, when we analyse the performance of our models and mistakes they make.

English data (228 individual LUs)			
# diff. LUs	# diff. fr.	prop. wrt. LUs	prop. wrt. all inst.
1 5 25	5 3 2	31 (13.60%)	513 (43.73%)
197	1	197 (86.40%)	660 (56.27%)
Data transl. from Ger. (316 individual LUs)			
# diff. LUs	# diff. fr.	prop. wrt. LUs	prop. wrt. all inst.
3 2 10 30	5 4 3 2	45 (14.24%)	1202 (51.21%)
271	1	271 (85.76%)	1145 (48.79%)
All data (434 individual LUs)			
# diff. LUs	# diff. fr.	prop. wrt. LUs	prop. wrt. all inst.
2 4 4 9 46	6 5 4 3 2	65 (14.98%)	1862 (52.90%)
369	1	369 (85.02%)	1658 (47.10%)

Table 3.12: English TRADR data: distribution of (non-)ambiguous LUs

Now, let us summarize the main properties of the English TRADR data. First, it belongs to the domain of team communication during disaster response. The domain has its influence on the language - most of the utterances are short and simple. Second, the data is multiclass and heavily imbalanced - 4,191 utterances are distributed between 190 various classes, and 137 of these classes have ten or less inhabitants. Third, more than half of all the data samples contain ambiguous targets, which may evoke up to six different frames. All this makes the process of training a frame classifier a very challenging task.

3.2 FrameNet data

In this section we will present more details about the FrameNet corpus. This is important, because if we want to use the corpus (or some part of it) to train a frame classifier that would work for TRADR data, we need to know the differences between the two corpora.

3.2.1 Domain

The FrameNet data includes sentences that were annotated using two different approaches: the so-called 'lexicographic' and 'full-text' annotations. The materials for

these two types of annotations come from different sources [Baker, 2008].

The sentences for lexicographic annotation come from the British National Corpus (BNC). According to Wikipedia [Wikipedia, 2020b], the BNC is a 100-million-word corpus of English. It is a result of the project that was started somewhere around 1990. About 90% of the BNC samples were taken from different newspapers, research journals from various academic fields, fiction and non-fiction books and many other types of texts. The rest of the corpus are samples of transcribed spoken data. The transcriptions represent conversations produced in various situations. It can be formal business or government meetings, conversations on radio shows or phone-ins. We do not know what part of the BNC was actually used for FrameNet annotations.

The data annotated according to the full-text approach comes from the American National Corpus (ANC), the Nuclear Threat Initiative website and the Wall Street Journal. The ANC consists of 22 million tokens of written and spoken data [Wikipedia, 2020a]. The project came to life in 1998. In the beginning the ANC included such corpora as the Indiana Center for Intercultural Communication Corpus (ICIC), the Charlotte Narrative and Conversation Collection (CNCC), as well as government documents and texts from a few U.S. publishers [Ide, 2008]. Later the corpus was extended with tweets and various web data.

The Wikipedia page about Nuclear Threat Initiative (NTI) says that it is a non-profit organization founded in 2001, which works to prevent catastrophic attacks and accidents with weapons of mass destruction and disruption – especially nuclear, biological, radiological, chemical and so on [Wikipedia, 2020e]. The data in the corpus obviously relates to these topics.

And finally, the Wall Street Journal (WSJ) corpus consist of data that belongs to the domain of business, finance and economics. More information about the corpus can be found in Paul and Baker [1992].

Again, it is unknown what parts of these corpora were taken for the full-text FrameNet annotations. But based on the corpora descriptions we can conclude that the FrameNet domain is very different from that of the TRADR data.

3.2.2 Semantic frames distribution

Before proceeding with the analysis of semantic frames distribution of FrameNet data, we would like to present more general information about the corpus.

So, with the help of NLTK FrameNet API [Schneider and Wooters, 2017] we extract from the FrameNet corpus the following information: sentences that were annotated with frames, positions of frame-evoking targets in the form of character indices, a lemmatized variant of each target (i.e. LUs) and frame labels themselves. With all duplicates deleted this makes up 199,508 sentences. In order to tokenize the sentences we use the same NLTK library for *Python* that we employed for the tokenization of TRADR data. In total our FrameNet data contains 4,751,140 tokens. This is about 209 times more than in TRADR corpus. An average sentence length counts approximately 23,81 tokens. This means that on the whole the FrameNet sentences are 3.5 times longer than TRADR ones.

There are 1,014 different semantic frames in FrameNet data, around 5.3 times more than in TRADR corpus. Because, like in case with TRADR, it is impossible to place all

1,014 frames into a single table or chart, in Table 3.13 we present only the information about the most frequent frames as well as the role of the least frequent ones in the FrameNet corpus. We chose such a format in order to be consistent with TRADR data visualization approach, and to make the distributions of frames in two corpora comparable.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Self_motion	6,453	3.24	27	1	0.01
Stimulus_focus	4,058	2.03	20	2	0.02
Emotion_directed	3,380	1.69	19	3	0.03
Clothing	3,181	1.59	11	4	0.02
Statement	3,037	1.52	12	5	0.03
Body_parts	2,672	1.34	8	6	0.02
Natural_features	2,361	1.18	9	7	0.03
Containers	2,111	1.06	12	8	0.05
Experiencer_obj	2,067	1.04	7	9	0.03
Judgment_communication	1,988	1.00	10	10	0.05
Total	31,308	15.69	135	598	0.30

Table 3.13: FrameNet data: frame distribution

If we take a look at the top ten most frequent frames shown in the left part of Table 3.13, we will notice at once that the given list is completely different from the list of the most frequent frames in TRADR data presented in Table 3.6. Taken together all instances of these frames make 15.69% of the data. This is less than the proportion of sentences belonging to the top ten most frequent frames in the TRADR corpus, but taken into consideration the fact that the FrameNet corpus is much larger and contains much more individual frames, it is still a lot. The right part of Table 3.13 shows the distribution of very infrequent frames, namely, how many separate frames have from one to ten instances, and what percentage of the data they make. We see that there are 135 different frames, each of which occurs ten or less times in the FrameNet data. All together they constitute a rather insignificant part of the data - only around 0.3%.

In general, the tendency in frame frequency distribution in FrameNet is similar to that of in TRADR. Top ten most frequent frames make less than 1% (in TRADR - about 5%) of total amount of different frames, but their instances compose quite a large part of the corpus. At the same time very infrequent frames make more than 13% (in TRADR - 72%) of all frames, and cover only a scanty amount of data. The difference in proportions of frequent and infrequent frames between FrameNet and TRADR can probably be explained by the fact that the frame frequency is relative with respect to the corpus size and the number of individual frames in the corpus. For instance, among the most frequent frames in TRADR data are frames that count about 100-200 instances, however, for FrameNet data such numbers mean only moderate frame frequency.

Our conclusion about the similarities in frame frequency distributions in both corpora is supported by the plots in Table 3.2. Two curves show how frame frequency changes depending on its rank, and they are comparable to the curves in Table 3.1.

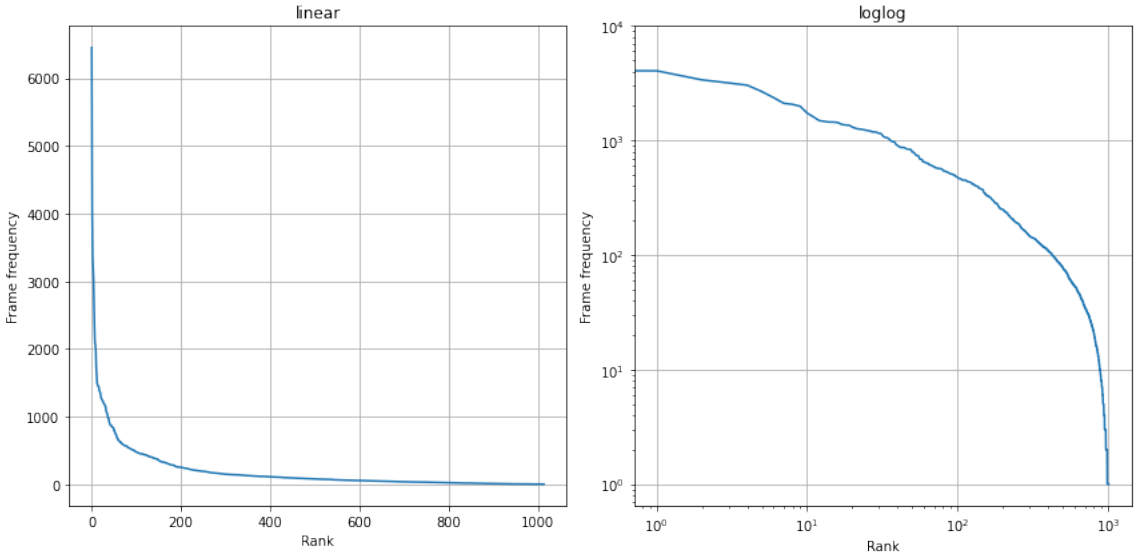


Figure 3.2: FrameNet data: frame frequencies

3.2.3 LU and POS distributions

Table 3.14 demonstrates the distributions of LUs and POS in the FrameNet corpus. Here we can notice some tendencies that are also observed in TRADR data. At the same time, there are also differences. We will discuss them below.

Top 10 most frequent LUs			LU POS distribution		
LU	# occ.	%	POS	# occ.	%
not.adv	544	0.27	.v	82,068	41.14
in.prep	472	0.24	.n	77,204	38.70
say.v	384	0.19	.a	33,759	16.92
tell.v	337	0.17	.prep	2,926	1.47
use.v	284	0.14	.adv	2,032	1.02
swim.v	278	0.14	.scon	752	0.38
people.n	274	0.14	.num	345	0.17
program.n	262	0.13	.art	267	0.13
eye.n	253	0.13	.idio	105	0.05
live.v	250	0.13	.c	50	0.03
Sum	3,338	1,67	Total	199,508	100

Table 3.14: FrameNet: top ten most frequent LUs and LU POS distribution

In total, the FrameNet corpus counts 8,333 individual LUs. The left part of the table shows the top ten most frequent LUs. First of all, we see that they are very different from the most frequent LUs in TRADR data shown in Table 3.9. The top two lines are occupied by an adverb and a preposition, the rest are verbs and nouns, while the list of top ten most frequent LUs in TRADR data contains nine verbs and one interjection.

All together the FrameNet instances that have as targets the most common LUs shown in Table 3.14 make only around 1.7% of all FrameNet data. Still, it can be

considered as a lot, because these ten frame-evoking LUs make approximately 0.12% of the total amount of different LUs. TRADR data reveals a similar trend.

The right part of Table 3.14 is also interesting. Here we can notice that the FrameNet LUs are distributed between 10 various POS tags, and more than 40% of all targets in the FrameNet corpus are verbs. Nominal targets take almost 39% of data thus forming the second largest group. The third largest group is represented by adjectival LUs. Other POS are much less frequent. In contrast to TRADR data, in FrameNet we can come across LUs that belong to such POS as subordination conjunction (.scon), coordinating conjunction (.c), article (.art) or idiomatic expression (.idio). Judging by this, we can conclude that the distributions of lexical units in both corpora are rather different.

3.2.4 Ambiguity of LUs

One more aspect we need to discuss in order to compare FrameNet and TRADR corpora is ambiguity of lexical units. This information is given in Table 3.15.

FrameNet data (8,333 individual LUs)				Top 10 ambiguous LUs	
# diff. LUs	# diff. fr.	prop. wrt. LUs	prop. wrt. all inst.	LU	# diff. fr.
2	10			strike.v	10
2	9			hit.v	10
5	8			order.n	9
9	7			make.v	9
17	6	1,301 (15.61%)	69,801 (34.99%)	development.n	8
34	5			swing.v	8
76	4			find.v	8
239	3			take.v	8
918	2			lose.v	8
7,031	1	7,031 (84.38%)	129,707 (65.01%)	develop.v	7

Table 3.15: Ambiguity in FrameNet data

In the left part of the table we see how many different LUs evoke certain number of different frames. The information is grouped exactly like in Table 3.12. We see that 7,031 LUs out of 8,333, i.e. more than 84%, are not ambiguous. At the same time there are LUs, though not many, that are highly ambiguous and evoke 7-10 various frames each. Note that in TRADR data the most ambiguous LUs evoke only 5-6 frames. It is interesting that the proportions of ambiguous LUs with respect to all individual LUs taken together are roughly the same in both corpora.

Now let us take a look at the proportion of instances with ambiguous targets with respect to the amount of instances. We can notice that in the FrameNet corpus there are more instances with non-ambiguous targets rather than with ambiguous ones, namely, around 65% against 35%. If we compare these numbers with those in Table 3.12, we will discover that in TRADR data it is the other way around: almost 53% of all instances contained ambiguous targets, and about 47% - non-ambiguous ones.

The right part of Table 3.15 demonstrates the top ten most ambiguous LUs in the FrameNet corpus. While in TRADR data all of them were verbal, here we can also

find two nominal LUs. In general, the most ambiguous LUs in FrameNet are absolutely different from those in the TRADR corpus. The only exception is the LU *'make.v'* which can be found in both lists.

To conclude, we would like to point out the following characteristics of the FrameNet corpus. First, it belongs to the domain of newspaper texts. As a result, its sentences are usually long (more than 23 tokens on average) and have a complex structure. Second, the FrameNet data, like TRADR is also multiclass and imbalanced. It also contains lots of instances with ambiguous LUs. We will summarize the similarities and differences between FrameNet and TRADR in a separate section below. This will help us use FrameNet data more efficiently in order to train a frame classifier for TRADR.

3.3 TRADR vs. FrameNet: comparison

The most obvious differences between the two corpora are, of course, the corpus size - FrameNet data contains about 209 times more tokens - and domain - FrameNet mostly consists of text samples that to a very large extent belong to such fields as business, politics and science, while TRADR data represents team communication in a disaster response scenario.

Both corpora are multiclass and highly imbalanced, but while TRADR data contains instances that belong to 190 classes (frames), FrameNet corpus, due to much bigger size, encompasses 1,014 different labels. 177 of 190 TRADR frames occur in FrameNet. This leaves 13 frames that do not have any instances in FrameNet, and 10 of these 13 frames were defined specially for TRADR (see Appendix B) and are absent in FrameNet at all, and 3 frames have only definitions but no actual instances in the FrameNet database. It should be mentioned that some of the 10 frames that were created specially for TRADR are very frequent, e.g., 'Communication_by_protocol' or 'Communication_response_message', others are rare, like 'Be_piece_of' or 'Being_reasonable'. Despite the fact that many of the frames are common for both FrameNet and TRADR, their roles are different, as the corpora have completely different domains.

Frame	# occ.	%
Capability	467	0.23
Motion	463	0.23
Perception_experience	589	0.30
Sending	214	0.11
Locative_relation	332	0.17
Create_representation	45	0.02
Identity	0	0.00

Table 3.16: Role of most frequent TRADR frames in FrameNet data

The domain difference is illustrated by Table 3.16, which shows the role of some of the most frequent TRADR frames in the FrameNet corpus. Most of them make only 0.1% - 0.2% of the whole number of instances. Frame *'Create_representation'* is rather infrequent - there are only 45 instances of this frame in FrameNet data. In

comparison, TRADR data while being much smaller in size contains 127 instances of *'Create_ representation'* frame. There is not a single example of *'Identity'* frame in the FrameNet corpus, despite the fact that the description of this frame exists in FrameNet online database.

Next, it is important to emphasize the difference in the lengths of instances of two corpora. While an average sentence in FrameNet data consists of almost 24 tokens, TRADR sentences are normally only about 7 tokens long. Such length divergence can be explained by domain differences, and this should be also taken into consideration if FrameNet data is to be used for training a frame classifier.

Naturally, FrameNet and TRADR also differ in the distribution of LUs. While FrameNet contains 8,333 individual LUs, TRADR has 434 ones, and only 280 of them occur in both corpora. Another noticeable difference between TRADR and FrameNet data is in the POS distributions of LUs. The reason for this is the fact that the motivations behind the choice of targets are completely different. While the main goal of FrameNet project is lexicographic, i.e. building a lexical database and studying the combinatorial properties of words [FrameNet, 2020], we try to capture the meaning of each utterance with respect to its relevance for the mission. So, in FrameNet verbal and nominal LUs are represented in more or less equal measure, and in general the LUs belong to a wider range of parts of speech. In TRADR we mostly concentrate on verbal LUs, as the dialogues are very task-oriented. In contrast to the FrameNet project, we are not interested in articles, numbers or conjunctions as targets. As a result, the LUs in focus, as well as frames they evoke, are very different in two projects. It is also obvious that semantic frame distributions differ a lot as well.

Despite differences in size and domain, the two corpora have some similarities. First, the shapes of the distributions of frame frequencies seem to be similar (compare plots in Tables 3.1 and 3.2). Second, the proportions of ambiguous LUs with respect to the total number of various LUs are approximately the same in both corpora.

Based on this information, we can conclude the following. Our main corpus, namely the TRADR corpus, is rather small. Taking into account the fact that the data is multiclass and highly imbalanced, the number of instances of some classes in it is clearly insufficient to train a reliable classifier. As the FrameNet data comes from a completely different domain and is imbalanced too, it is unlikely that a frame classifier trained on FrameNet will perform well on TRADR data. However, the FrameNet data can be helpful as a source of additional training examples for the classifier. We will check these hypotheses in the next chapter.

3.4 German TRADR and SALSA data

As we will also train a German version of the frame classifier, it is necessary to introduce our German data, namely the German TRADR corpus and SALSA data. But first we would like to explain why we need to present German TRADR data separately from the English one, and why it was annotated with semantic frames from scratch instead of a simple transferring of frame labels from the translated utterances to the original German ones.

To start with, only six dialogues out of nine were translated, which means that

three dialogues needed to be annotated anyway. Second, the original utterances differ from their translations in structure, word order, length and so on, because usually it is impossible to perform word by word translation. Finally, we wanted to sample additional data from the German SALSA corpus, which was annotated with frames that slightly differ from those used in the English FrameNet. For all these reasons we had to perform the annotations of all nine German TRADR dialogues separately, and as a result there is no one-to-one correspondence in frame labels between German utterances and their translated counterparts.

It is important to point out the differences in annotation approaches to English and German dialogues. One of them is that the German TRADR data was not annotated with frame-evoking targets and, consequently, with lexical units. Instead, the frame choice for each utterance was justified by the so-called *'target related elements'*, which represent the whole phrase that the target is a part of. Often, the target is the head of the phrase, but not necessarily. So, e.g., all tokens in the utterance *"Erstes Geschoss bei der Brandentwicklung"* are considered to be target related elements of *'Locative_relation'* frame, while according to the annotation approach for English data, only the preposition *'bei'* (namely, its English counterpart) would be marked as the target. It should be noticed that using target related elements instead of simple targets has pros and cons. One of the advantages is that target related elements provide more context and may be helpful for frame recognition in case of elliptical utterances, when the targets themselves are omitted. The downside of the approach is that it can be difficult for the classifier to decide which of the target related elements it should focus on to assign the right frame.

Because in the German data targets were replaced with target related elements, elliptical utterances were also treated differently. While such instances in the English TRADR were excluded from the experiments, similar TRADR instances in German were included into training/test data as long as they had at least one target related element. Notice that it is still possible to recognize the elliptical utterances in the German data - all omitted frame related elements that are important for understanding the meaning of the utterance were restored in a separate column during the annotation.

We wanted to make German frame annotations compatible with a set of frames used for the annotation of the SALSA corpus (we will present it later in this section), because this would enable sampling additional training instances from it. In order to do so, we had to replace some of the frames used for the annotation of English TRADR dialogues with their SALSA-typical variants. This was not an easy task. According to Burchardt et al. [2006], the SALSA corpus relies on an older version of FrameNet, namely FrameNet 1.2, but because of differences between English and German, certain FrameNet frames required adaptation for the SALSA annotation. First, frame elements of some frames got merged because of ontological distinctions between their roles in English and German. Second, new frame elements were introduced for some frames. Third, some non-lexical FrameNet frames (i.e. general frames that do not have instances) were used as lexical. And finally, new frames were introduced. Unfortunately, we were not able to find any official definitions of the SALSA frames, and do not know for sure if the names of the adapted frames were somehow changed or not. In addition to this, the FrameNet version that was used for the annotation of the English TRADR dialogues, namely FrameNet 1.7, differs from FrameNet 1.2 - some new frames were added, some old

frames got split or merged.

Trying to establish a mapping from FrameNet 1.7 to SALSA, we noticed that some frame labels were identical in both corpora, e.g., *'Activity_ start'* or *'Activity_ ongoing'*, so we left them as they were. When we had both English and German labels with identical usage examples, we replaced the English label with the German one, e.g., *'Closure'* became *'schliessen5-salsa'*, and *'Needing'* - *'brauchen2-salsa'*. Moreover, we found out that certain frames in FrameNet 1.7 correspond to two different ones in SALSA. We replaced them accordingly, e.g., instead of *'Being_ operational'* we used *'ausfallen1-salsa'* and *funktionieren1-salsa'*, and instead of *'Change_ operational_ state'* - *'starten1-salsa'* and *'ausschalten1-salsa'*. For frames that did not have SALSA-equivalents we used labels from FrameNet 1.7. Finally, we need to mention that in contrast to FrameNet the relations between frames are not annotated in the SALSA corpus. Because of this, only frame labels coming from FrameNet were annotated with parent frames.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Communication_by_protocol	731	20.77	31	1	0.88
Communication_response_message	195	5.54	22	2	1.25
Presence	183	5.20	12	3	1.02
Grasp	168	4.77	10	4	1.14
Capability	156	4.43	6	5	0.85
Motion	155	4.40	8	6	1.36
Sending	140	3.98	1	7	0.20
darstellen3-salsa	106	3.01	4	8	0.91
Perception_experience	105	2.98	2	9	0.51
Scrutiny	99	2.81	9	10	2.56
Total	2,038	57.91	105	376	10.68

Table 3.17: Original German TRADR data: frame distribution

Due to the differences in the annotation approach that have been listed above, the total number of frame labels used for the annotation of the original German TRADR data is less than the total number of frames used to annotate their translations: 152 and 162 labels, respectively. The labels are distributed across 3,519 instances. Table 3.17 shows this distribution in detail. As usual, the left part of the table presents the top ten most frequent frames, and the right part illustrates the role of infrequent frames in the corpus. If we compare this table with the corresponding table presenting the distribution of frames in translated data (see Table 3.6), we can notice that two distributions are very similar. First, the frame labels and their order in the left parts of both tables are almost the same with two exceptions: the frames *'Presence'* and *'Grasp'* are absent in Table 3.6, which contains the frames *'Identity'* and *'Locative_ relation'* instead. Note that *'darstellen3-salsa'* corresponds to the *'Create_ representation'* frame. Second, the most frequent as well as infrequent frames make approximately the same proportions in both datasets. In the original German dialogues their share is about 58% and 11% respectively, in the translated data - about 59% and 13%.

Now let us proceed with the SALSA corpus. The corpus extends the German treebank called TIGER with semantic annotations, i.e. given a syntactic tree, its root was annotated with the frame label, and the root's edges - with frame elements' labels.

The annotation focused on verbal targets in the first place [Burchardt et al., 2006]. As TIGER treebank consists of articles from the German newspaper *'Frankfurter Rundschau'* [Brants et al., 2002], the domain of SALSA data is close to that of FrameNet. We extract sentences, targets and the corresponding frame labels from SALSA with the help of SALSA API, which is available for downloading at SALSA Project [2020]. We do not have any information about lexical units or parent frames, as they are not annotated in SALSA. In total we have 35,236 sentences (contain 838,307 tokens) that belong to 880 different frames. Only 80 frames are common with the German TRADR corpus, which includes 152 individual frames.

The frame distribution in SALSA data is presented in Table 3.18. We see that the top ten most frequent frames listed in it are different from those in both German TRADR and FrameNet corpora. Obviously, frame distributions in general are also different in these three datasets. Still, it is clear that SALSA data, like TRADR and FrameNet, is highly imbalanced - the instances that belong to the top ten most frequent frames make up about 29% of all the data. One can also notice that SALSA contains a lot of frames that occur only once or twice - together they make up almost 29% of all the available frames. And if we consider all the frames that have ten or less instances, we will end up with 509 frames which compose more than 57% of all labels but cover only 5% of data. In comparison, in FrameNet such frames make up only slightly more than 13% and cover 0.30% of data.

Top 10 most frequent frames			Role of infreq. frames in the data		
Frame label	# occ.	%	# frames	# occ. per fr.	%
Calendric_unit	1,854	5.26	148	1	0.42
Telling	1,608	4.56	104	2	0.59
People	1,593	4.52	61	3	0.52
Political_locales	1,380	3.92	49	4	0.56
Request	780	2.21	37	5	0.53
Statement	719	2.04	29	6	0.49
Support	649	1.84	27	7	0.54
Judgment_communication	601	1.71	21	8	0.48
Leadership-fnsalsa	542	1.54	13	9	0.33
Causation	472	1.34	20	10	0.57
Total	10,198	28.94	509	1,768	5.01

Table 3.18: SALSA data: frame distribution

So, we see that basically everything that what written earlier about the similarities and differences between the English TRADR data and FrameNet can be used to compare the German TRADR data and SALSA - the two corpora differ in domain, size and distribution of semantic frames. In the next chapter we will check whether and how the SALSA corpus can be used to improve the performance of the frame classifier.

3.5 Data for experiments

In what follows we will describe how we prepared and partitioned our data for the experiments that are presented in the next chapter. We will start with the data in

English.

Before deleting duplicates, there were 3,521 utterances in English TRADR and 200,750 sentences in FrameNet. We consider duplicates the instances that have matching sentences, targets and frame labels. After deleting them, we were left with 2,930 TRADR instances and 199,509 FrameNet ones.

We split both corpora into main data, which is used for training and validation, and makes up 90% of each dataset, and test data, which makes up 10%. So, the main part of the TRADR data counts 2,637 instances, the part for testing - 293. The main part of FrameNet has 179,559 samples, and the corresponding test data - 19,950. As both corpora contain multiple frame labels that are represented by a single instance each, the main part of each corpus has frames that are not present in the part reserved for testing, and vice versa. So, after the split the main parts of TRADR and FrameNet have 184 and 1,012 individual frames, in comparison with 190 and 1,014 frames before the split. As we plan to perform 5-fold cross-validation using the main parts of the two datasets, we need to remove the frames that have less than 5 instances each, otherwise it will be impossible to split the main parts into 5 folds, so that all frames are represented in each fold. After the rare frames removal, we have 81 labels in TRADR data and 931 labels in FrameNet. The corpus sizes decrease down to 2,444 and 179,386 sentences respectively.

5-fold cross validation assumes that during the training the main data will be randomly split into five equal folds (parts), of which four will be used for training and one for validation. Thus, in case of TRADR in each iteration the training data will include 1,955 samples, and the validation data - 489. For FrameNet it will be 143,509 and 35,877 samples, respectively. Rotating the parts as shown in Figure 3.3 we will train five models. The model with the best validation IBA score will be evaluated on the test sets.

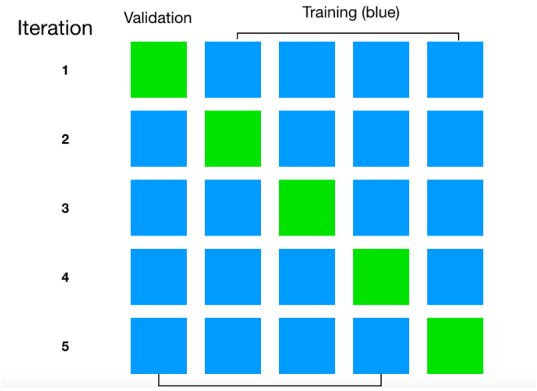


Figure 3.3: 5-fold cross-validation procedure (taken from Harlan [2020])

Next, we remove from both test sets the instances of the frames that are absent in the corresponding main parts, because the frame classifier will not be able to learn such frames, and it would be unfair to test it on their instances. As a result we are left with 268 utterances in TRADR test set (test set A) and 19,923 sentences in FrameNet test set. in addition to this, we create two more TRADR test sets (B and C), which are subsets of the test set A. The test set B contains the same frames as set A, except for

'Communication_ by_ protocol' and *'Communication_ response_ message'* frames, and counts 247 samples. We remove the above-mentioned frames, because most of them represent short stable expressions, like *"Team leader for operator one"*, or *"Roger"*, and therefore are easy to learn and predict. As these frames are among the most frequent, they may lead to biased evaluation results for some metrics. While the test sets A and B are designed for testing the models trained only on those samples whose labels are present in TRADR data, one of our classifiers is to be trained purely on FrameNet data. We know that some of the TRADR frames are absent in FrameNet. So, in order to have a universal test set that could be used for evaluation of any model, we built the test set C. It counts 234 instances of frames that occur in both TRADR and FrameNet main parts. All these data splits and manipulations are summarized in Table 3.19. Parts of the data that are actually used for training are given in bold.

Data	TRADR	# frames	FrameNet	# frames
Size before dropping duplicates	3,521	190	200,750	1,014
Size after dropping duplicates	2,930	190	199,509	1,024
Main (training + validation) data size	2,637	184	179,559	1,012
Main data size (rare frames removed)	**2,444**	**81**	**179,386**	**931**
Training data size	1,955	81	143,509	931
Validation data size	489	81	35,877	931
Test data size (initial)	293	-	19,950	-
Test data size (rare frames removed)	268	81	19,923	931
Test set A	268	81	-	-
Test set B	247	79	-	-
Test set C	234	50	-	-

Table 3.19: English TRADR and FrameNet training, validation and test data

Similar manipulations are performed with the German TRADR data. Because we are planning to use SALSA only for sampling and not as the main source for training any models, we do not partition it. So, first we remove all duplicates from our German TRADR corpus. As a result we have 2,813 utterances. Here we need to point out that we identify the duplicates by the identical utterances, frame labels and target related elements. Notice that SALSA data does not contain any duplicate sentences. Next, the data is split into main and test parts, rare frames are removed, and the frame labels in test data are brought in accordance to those in the main part to form the test set A. The test set B is built from the test set A as described earlier. As we will use the 5-fold cross-validation procedure also for training the German frame classifier, the main data is split into training and validation parts as described above. All these steps are shown in Table 3.20.

Before presenting our experiments and their results, we would like to sum up the main accomplishments of this chapter. We have presented three different corpora annotated with semantic frames, namely TRADR, FrameNet and SALSA. We have thoroughly examined and compared their domains, size, distribution of semantic frames and (where it was possible) the distributions of lexical units and their parts of speech. We have

tried to predict how certain characteristics of our data may influence the performance of the frame classifier, and what effect the differences between the corpora may have on its performance on the TRADR corpus in case out-of-domain data (FrameNet or SALSA) is used to train the model. We will check these assumptions in the next chapter.

Data	TRADR	# frames
Size before dropping duplicates	3,519	152
Size after dropping duplicates	2,813	152
Main (train. + valid.) data size	2,532	146
Main data size (rare frames removed)	**2,378**	**72**
Training data	1,902	72
Validation data	476	72
Test data size (initial)	281	-
Test set A (rare frames removed)	259	72
Test set B	217	70

Table 3.20: German TRADR training, validation and test data

Chapter 4

Experiments and Discussion

In this section we will present semantic frame classifiers for both English and German TRADR dialogues. Our main focus will be on English data. We will introduce several models that will be split into three groups.

The first group will include three initial models, namely a baseline model, which is a pretrained $\text{BERT}_{\text{BASE}}$ model fine-tuned with a standard approach on TRADR data for the task of sentence classification, and two $\text{PAFIBERT}_{\text{BASE}}$ models [Tan and Na, 2019] that we re-implemented. One of the PAFIBERT models is trained on FrameNet data, as presented in the paper by [Tan and Na, 2019], the other, which we will call a basic model in order to differentiate it from the baseline, is trained purely on TRADR data. We will analyse the mistakes that these classifiers make and check how well a model trained on data coming from one domain can perform on data from a completely different one.

Next, we will investigate several modifications of the basic model, and discuss their performance. The modifications will comprise changing the size of the context window around the frame-evoking target, adding samples from FrameNet to TRADR training data, extending BERT embeddings with extra features and applying frame filtering. These models will be included into the second group.

Finally, our basic classifier for German which is trained on TRADR data and also relies on $\text{PAFIBERT}_{\text{BASE}}$ will be introduced. As in the case of the frame classifiers for the English TRADR dialogues, we will study the effect of sampling and adding features on our German model, and try to explain why and where the classifier succeeds or fails. We will not consider the modifications of the context window size, because instead of targets German TRADR dialogues are annotated with frame related elements, which often include several tokens, so that it is difficult to compare the window sizes and the corresponding effect across German and English data. Frame filtering, which also assumes having targets/lexical units annotation, will be omitted as well.

4.1 Initial models for English TRADR

This section will be devoted to the first group of the frame classifiers, which includes three models. We will start with the architecture of these models, then we will present the results on the corresponding test sets and discuss the mistakes.

4.1.1 Architecture

As a baseline we use the *BertForSequenceClassification* model from the *Transformers* library by *HuggingFace* [Wolf et al., 2020]. We choose it as the most straightforward way to perform sequence classification. We do not consider frame classifiers reported in Chapter 2 as possible baselines, because they were trained on text corpora, and the domains of these corpora are very different from ours. So, *BertForSequenceClassification* is a pretrained $\text{BERT}_{\text{BASE}}$ model with an additional linear layer on top of the pooled output. The pooled output is the last layer hidden-state of the first token ([CLS]) of each sequence in a batch, and has shape $b \times h$, where b is the batch size and h is the hidden layer size. The size of the additional linear layer depends on the number of labels we provide to the instantiation of the *BertForSequenceClassification* model. We use 81 frame labels as explained in Section 3.5 and fine-tune the model on the main part of the English TRADR data (2,444 utterances) with 5-fold cross-validation. The training is performed for 8 epochs per fold using an adaptive learning rate that starts with 3e-5 and an *AdamW* optimizer.

Now let us illustrate the architecture of PAFIBERT with its position-based attention mechanism, and present two frame classifiers which rely on it.

The $\text{PAFIBERT}_{\text{BASE}}$ model expects as input a sequence (an utterance or a sentence) in the form of BERT token IDs, attention masks for the given sequence that serve to distinguish between the original and padded tokens, and a position vector encoding where the frame-evoking target is located in the sequence. The position vector has the same length as the vector with input IDs and consists of zeros and ones with ones marking the tokens that are part of the target as shown in Figure 4.1. Note that if the BERT tokenizer splits a token into smaller sub-tokens, each sub-token is considered to be a part of the target, and is marked accordingly. We can also provide the model with token type ids (also known as segment embeddings) and positional embeddings, or, because our input only consists of single utterances/sentences, we can let BERT pre-compute them for us.

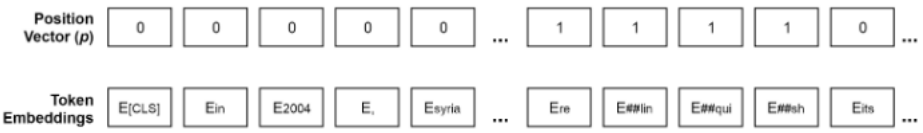


Figure 4.1: PAFIBERT: input tokens and the corresponding position vector (taken from Tan and Na [2019])

All required inputs are then fed through the 12 transformer blocks of $\text{BERT}_{\text{BASE}}$ model to produce a matrix of hidden states H of size $n \times d$, where n is maximal sequence length and $d = 768$ is the hidden layer size. As Tan and Na [2019] do not say explicitly the hidden state of which encoder block they use, we assume that H is the hidden state of the last block.

In contrast to *BertForSequenceClassification*, instead of a single linear layer on top of standard $\text{BERT}_{\text{BASE}}$ before the classification is performed, $\text{PAFIBERT}_{\text{BASE}}$ has an additional position-based attention layer. The purpose of this layer is to attend to the target and the context around it. While the *BertForSequenceClassification* model has

no clue what tokens in the sequence are important for the right frame assignment, the PAFIBERT model utilizes the information about the target, and thus is able to produce much better results.

The attention mechanism in PAFIBERT uses H to build a vector $[c;t]$, which is a concatenation of a context vector c and a target vector t. The d-dimensional context vector

$$c = H^T\alpha \tag{4.1}$$

captures a relation between the hidden representations of the input sequence and an alignment vector α of length n, which contains weights showing how much each token in the input is important as a context element. In order to determine which tokens are parts of the context and which ones should be ignored, a context window around the target token is used with lower and upper bounds defined as

$$\beta_1 = \begin{cases} p_{start} - w & \text{if } (p_{start} - w) > 1 \\ 1 & \text{if } (p_{start} - w) \leq 1 \end{cases} \qquad \beta_2 = \begin{cases} p_{end} + w & \text{if } (p_{end} + w) < n \\ n & \text{if } (p_{end} + w) \geq n \end{cases} \tag{4.2}$$

where p_{start} and p_{end} are the smallest and the largest target positions in the sequence, and w is the window size. If the target is a single token, then $p_{start} = p_{end}$. We keep $w = 10$ used by Tan and Na [2019] for both of our classifiers reusing PAFIBERT. The alignment weights for all the tokens that lie outside of $[\beta_1, \beta_2]$ range are set to zeros, and for tokens that are inside of it, the weights are calculated using the *Softmax* function:

$$\alpha_i = \frac{exp(h_i t)}{\sum_{j=\beta_1}^{\beta_2} exp(h_j t)} \tag{4.3}$$

where h_i is a hidden representation of a token within $[\beta_1, \beta_2]$ and t is the hidden representation of the target(s). So, with all weights summing up to one, the context vector c is just a weighted average of the input sequence.

The d-dimensional target vector t

$$t = H^T p \tag{4.4}$$

captures a relation between H and the position vector p, which is part of the input together with the sequence itself. As mentioned earlier, the position vector indicates where the targets in the sequence are.

Finally, the concatenated context and target vectors $[c;t]$ go through an additional linear layer W_c and a *Tanh* activation function to produce a d-dimensional attentional hidden state $\tilde{h}$. Next, $\tilde{h}$ is fed into a standard linear layer W_s with a *Softmax* function on top to perform classification. The whole pipeline with all intermediate steps is well illustrated in Figure 4.2.

We reuse this architecture to build two frame classifiers, which we train on different datasets: one - entirely on FrameNet (main part with 179,386 instances), the other - entirely on the English TRADR data (same as used for fine-tuning). As FrameNet and TRADR differ in size and in the number of frame labels, the classifier trained on FrameNet data is able to differentiate between 931 frames, while the classifier trained on TRADR can distinguish between 81 classes. Due to the domain differences, the

maximum input length the classifiers can process also differs. It is 314 tokens for the former, and 45 tokens for the latter. In all other aspects two models are absolutely identical. The training procedure is the same that was used for training the baseline model.

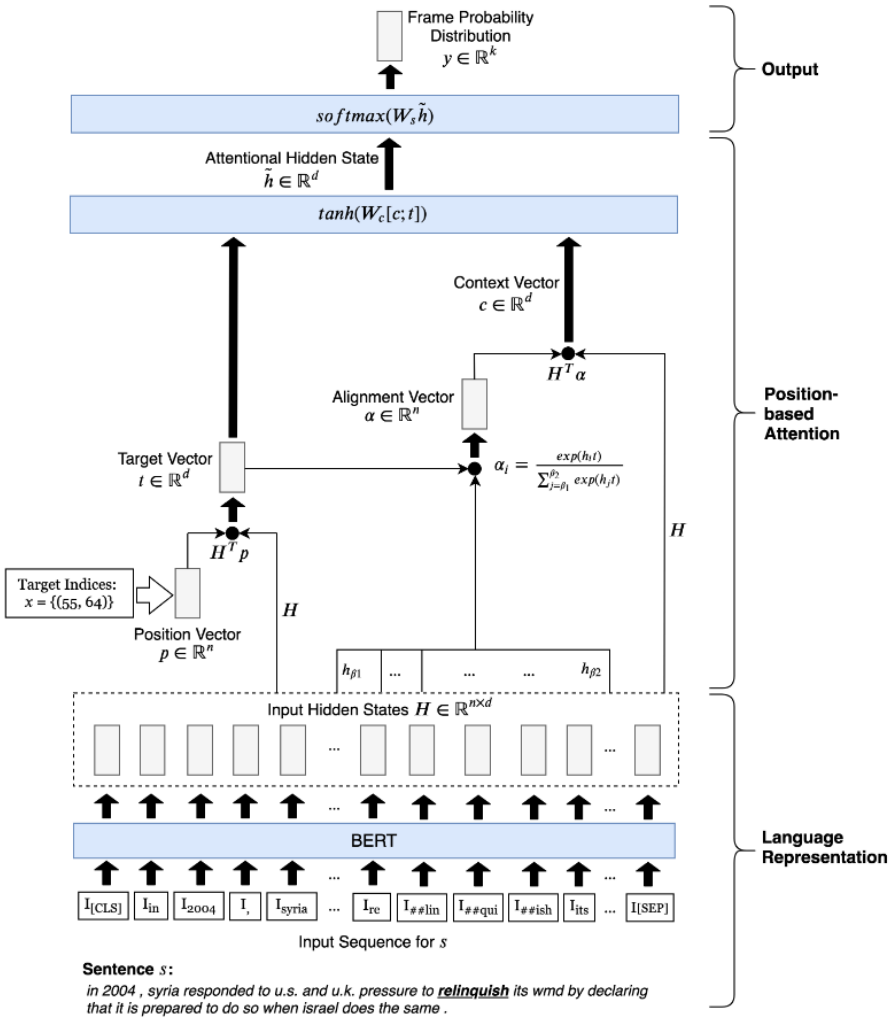


Figure 4.2: PAFIBERT$_{\text{BASE}}$ architecture (taken from Tan and Na [2019])

4.1.2 Results and discussion

Now let us have a look at the performance of our initial models and discuss the results. For evaluation we use a *Python* package called *imbalanced-learn* developed by Lemaître et al. [2017]. It allows to produce a classification report using such metrics as precision (PRE), sensitivity (SEN), specificity (SPE), F-score ($F1$), geometric mean (GM) and the index of balanced accuracy with $\alpha = 0.1$ ($IBA_{0.1}$). All metrics are calculated using macro-averaging. The report also contains the information about the test set size (SUP). As precision and F-score are considered sensitive to imbalanced data, we won't rely on them during the discussion. We won't remove them either to let the interested reader compare the scores. As mentioned in Chapter 2, we will refer to IBA as our main performance measure. The evaluation results are presented in Table 4.1.

The baseline is tested on a part of the English TRADR data unseen during the training, namely on the test sets A, B and C. More details about the data can be found in Section 3.5. From Table 4.1 we see that the baseline model demonstrates rather unsatisfactory performance - the IBA is only 32% - 37% depending on the test set, and other metrics (except specificity) have very low scores too. The reason for this is the fact that simple fine-tuning does not integrate information about the frame-evoking targets and their contexts, so that it is impossible for the model to guess what tokens in the sequence it has to focus on. So, it is obvious that in order to improve the performance, we need to tell the model which tokens in each utterance it should pay attention to, and the PAFIBERT model suggested by Tan and Na [2019] provides a convenient way to do so.

We test the PAFIBERT classifier trained on TRADR data, on the test sets A, B and C, while the other classifier trained on FrameNet is tested on both FrameNet test set and test set C. The latter represents a convenient way to compare the performance of the models trained on in- and out-of-domain data, as it contains only the instances of frames common for both FrameNet and TRADR.

Let us first have a look at the performance of the PAFIBERT model trained on the FrameNet data. This classifier has the IBA of 91% when evaluated on the data coming from the same distribution, however, when tested on TRADR data, it shows much worse results, namely, only 51% IBA. We can notice that the main reason for low IBA score is the bad sensitivity (recall). Such results support the hypothesis that the model is very domain-specific, it does not generalize well, and we cannot simply re-use it for TRADR data without any modifications or further fine-tuning. At the same time this classifier is still better than the baseline. So, we see that attending to the right tokens is really crucial for the performance.

Classifier	Test set	PRE	SEN	SPE	$F1$	GM	$IBA_{0.1}$	SUP
Baseline	TRADR set A	0.33	0.39	0.96	0.35	0.50	0.37	268
	TRADR set B	0.29	0.35	0.96	0.30	0.46	0.32	247
	TRADR set C	0.30	0.35	0.96	0.31	0.46	0.32	234
PAFIBERT trained on FrameNet	FrameNet set	0.92	0.92	1.00	0.92	0.96	0.91	19,923
	TRADR set C	0.71	0.53	1.00	0.58	0.63	0.51	234
PAFIBERT trained on TRADR	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.1: Experiment results

Before discussing the performance of the PAFIBERT model trained on TRADR data, we would like to have a closer look at the mistakes that the classifier under consideration makes when evaluated on test set C, and understand why exactly it happens. This can be relevant for our further experiments with sampling from FrameNet described in Section 4.2.2.

So, we know that FrameNet and TRADR have different frame distributions, and, as discussed in Section 3.3, if we take any of the most frequent TRADR frames, its

proportion in the FrameNet corpus will be maximum 0.3%, i.e. such a frame will be represented by less than 600 instances. This number does not look terribly small, taking into consideration that the whole TRADR training corpus contains only 2,444 samples unevenly distributed between 81 frames.

To get a better understanding of why the classifier trained on FrameNet fails on the TRADR test set C, let us study more precisely the distribution of frames from our test set in the FrameNet training data. The TRADR test set C contains instances that belong to 50 different frames. All these frames are present in the part of FrameNet data reserved for training. Taken together, they count 14,517 instances and make up around 8% of FrameNet training data. The rarest frame is *'Holding_off_on'*, which has only 12 instances in the FrameNet training data, the most frequent is *'Awareness'*, represented by 1219 instances. On average, each of the 50 frames in the TRADR test set C appears 290 times in the FrameNet training set. This seems to be enough to learn them. And still, the IBA score was only slightly better than 50%.

Let us have a look at the mistakes made by the classifier. Table 4.2 shows the top ten frames from the TRADR test set C that were incorrectly classified. The second column shows the number of misclassified instances with respect to the total amount of instances of the given frame in test set. The third column shows the number of training FrameNet instances.

True frame	Error rate	# training inst.
Capability	16/35 (46%)	422
Create_representation	12/12 (100%)	39
Locative_relation	12/15 (80%)	299
Motion	7/21 (33%)	409
Being_obligated	6/6 (100%)	37
Perception_experience	5/25 (20%)	529
Possibility	4/5 (80%)	51
Scrutiny	3/4 (75%)	709
Being_located	3/3 (100%)	96
Awareness	2/4 (50%)	1219

Table 4.2: Top 10 frames from the TRADR test set C incorrectly classified by the PAFIBERT model trained on FrameNet data

We see that all frames having less than a hundred of training instances demonstrate a very high error rate - almost 100%. However, in many cases the number of training instances is much larger, but the error rate is still high, e.g., this is true for *'Capability'*, *'Locative_relation'* or *'Scrutiny'* frames. Obviously, there are reasons for such an unsatisfactory classifier performance, other than an insufficient number of training examples. In what follows we will try to summarize these reasons.

First of all, some frames by definition are really semantically close to each other, e.g., *'Capability'*, *'Possibility'* and *'Likelihood'*, or *'Scrutiny'* and *'Inspecting'*, or *'Awareness'* and *'Opinion'*. In most cases neither the target nor the context can help differentiate the nuances of meaning of such frames.

Next, due to the fact that FrameNet is very fine-grained, many TRADR instances got classified as belonging to very specific frames which we did not use when annotating the TRADR data, like *'Interior_profile_relation'* and *'Non_gradable_proximity'* (we use their parent frame *'Locative_relation'* instead), or *'Self-motion'* instead of more general *'Motion'* frame.

Another reason for poor performance of the classifier in question is the fact that TRADR instances of certain frames have targets that, due to domain differences, are not typical for these frames in FrameNet. For instance, all TRADR samples of *'Create_representation'* frame were misclassified, because the model expected *'draw'*, *'carve'* or *'sketch'* as targets, but got *'take/make a picture'* and labeled the input utterances as *'Physical_artwork'* instead. A similar situation is observed in case of *'Being_located'*, *'Presence'* and some other frames.

Of course, there is also a problem of ambiguity. For example, the target *'change'* can evoke both *'Replacing'* and *'Cause_change'* frames, and the target *'lie'* - *'Posture'* and *'Being_located'*.

Now let us have a look at the test set IBA scores obtained by the PAFIBERT model trained on the TRADR data. We can notice that they are slightly lower than the in-domain test set IBA score of the counterpart model trained on FrameNet data, namely 87% - 88% vs. 91%. We attribute it to the fact that the latter model was trained on a much larger dataset. It is worth mentioning that despite the huge difference in the size of training data, the difference in IBA scores is only about 3% - 4%, which allows us to hypothesise that the PAFIBERT model does not really need a lot of training examples to achieve relatively good accuracy. If we compare the results of both PAFIBERT-based models on the test set C, we can once again see that the in-domain model (even if it is trained on much smaller data) obtains much higher IBA scores than the out-of-domain one. Finally, we can clearly see that the PAFIBERT model trained on TRADR is much better than the baseline. As it turned out to be the best among the three classifiers, from now on we will consider it to be the basic model.

Also, we see that in case of our basic model the results for the TRADR test sets A, B and C look very similar with only a slight performance decrease for smaller subsets. This can probably be explained by the fact that our metrics are not sensitive to class sizes, so when we remove two majority classes, it does not influence much the overall result. However, this is not quite the case with the baseline. Here, the performance decrease is more visible. Our explanation is as follows. Classes *'Communication_by_protocol'* and *'Communication_response_message'* removed from the TRADR test sets B and C are probably the ones that the baseline can predict better than other frames, due to the fact that these two classes in most cases are represented by very short utterances like *"Yes"*, *"Roger"* and so on. Because the baseline does not possess any information about the targets, the longer the sentence is, the more likely the model will misclassify it. So, when the above mentioned classes get removed, the performance inevitably deteriorates.

Finally, we can observe that regardless of the model and test set, the specificity is perfect. This can be explained by the fact that macro-averaging, used by *imbalanced-learn* to calculate the scores, by definition relies on confusion matrices built according to the one-against-all approach. If we consider a single matrix, the only thing that matters for specificity is that the whole lot of instances of classes that are not currently

in focus (i.e. negative classes) should not be classified as positive. The true classes of such instances, and how well the classifier can distinguish between them are not important for specificity. So, if the number of false positives is close to zero, the specificity will always be very high.

So, in this section we have discussed the performance of our baseline model and two frame classifiers both relying on the PAFIBERT architecture, but trained on different data (FrameNet and TRADR). Judging by the results we can make the following conclusions. First, the information about the frame target and its context is crucial for the correct frame recognition. Second, both PAFIBERT models demonstrate rather good results when tested on in-domain data. In addition, it looks like the PAFIBERT classifier actually does not require very much data to achieve good scores - trained on only 2,444 TRADR instances it achieves the IBA of about 86% - 88% on unseen data. Second, the performance of the FrameNet-based PAFIBERT model on TRADR data is far from being perfect. The analysis of mistakes shows that it happens not only because of the difference in domains and the ambiguity of targets, but also because of different annotation approaches and small nuances in frame meanings that are especially difficult to detect. We do not exclude the possibility of further modifying/fine-tuning of the FrameNet-based PAFIBERT model on TRADR data, but this lies out of the scope of this work. However, we will investigate the effect of sampling from FrameNet in one of the later sections.

4.2 Adjustments of the English PAFIBERT model

Aiming at performance improvement, in what follows we will present several modifications and adjustments of the PAFIBERT model trained on TRADR. We will discuss the results of these experiments, and analyse the typical mistakes made by the classifiers.

4.2.1 Changing context window size

Following Tan and Na [2019], who found the window size $w = 10$ to be optimal for the FrameNet data, we used this window size in both our PAFIBERT-based models presented in the preceding section. However, as was mentioned earlier, the FrameNet and TRADR datasets have different average sentence lengths, namely, 24 and 7 tokens, respectively. So, even if we take into consideration that some tokens are split by the BERT tokenizer, $w = 10$ seems to be excessive for TRADR data, as in most cases it encompasses the whole utterance.

In order to check whether a smaller window size can help increase performance, we retrained the basic model using $w = 1$, $w = 3$ and $w = 5$. The results of these experiments are presented in Table 4.3. The last row of the table contains the test scores of the basic model (has $w = 10$), which are given for reference. We see that our hypothesis about a smaller window size being better for TRADR data was not confirmed. Moreover, judging by the scores, there seems to be no direct correlation between the window size and the performance. E.g., $w = 1$ gives higher scores than $w = 3$ or $w = 5$, and $w = 10$ provides better results than $w = 1$. It looks like the fluctuations in performance do not follow any stable pattern.

Window size	Test set	PRE	SEN	SPE	$F1$	GM	$IBA_{0.1}$	SUP
$w = 1$	TRADR set A	0.89	0.88	1.00	0.88	0.93	0.87	268
	TRADR set B	0.90	0.88	1.00	0.88	0.92	0.87	247
	TRADR set C	0.90	0.88	1.00	0.88	0.92	0.87	234
$w = 3$	TRADR set A	0.90	0.87	1.00	0.87	0.92	0.86	268
	TRADR set B	0.90	0.87	1.00	0.87	0.92	0.86	247
	TRADR set C	0.90	0.86	1.00	0.87	0.91	0.85	234
$w = 5$	TRADR set A	0.88	0.88	0.99	0.88	0.92	0.87	268
	TRADR set B	0.88	0.87	0.99	0.87	0.92	0.86	247
	TRADR set C	0.88	0.87	0.99	0.87	0.91	0.85	234
PAFIBERT trained on TRADR (basic model)	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.3: Performance of the basic model with different window sizes

To have a better understanding of what is going on, we compared the validation IBA scores for different window sizes and folds during the 5-fold cross-validation process. The scores are presented in Table 4.4. The best of them are given in bold. We see that regardless of window size, the models were able to reach approximately the same IBA, namely, 90 - 91%. Moreover, the validation IBA score averaged across five folds is the same for all of them.

Fold	$w = 1$	$w = 3$	$w = 5$	$w = 10$
Fold 1	**0.91**	0.89	0.90	0.89
Fold 2	0.88	0.89	0.88	0.89
Fold 3	0.86	0.89	0.90	**0.90**
Fold 4	0.90	0.87	0.87	0.89
Fold 5	**0.91**	**0.91**	**0.91**	**0.90**
Avg.	0.89	0.89	0.89	0.89

Table 4.4: The basic model: IBA for different folds and window sizes

A possible explanation of this situation can be the fact that we have too little training and test data in combination with the stochastic nature of the learning algorithm. Probably, having more data would result in scores which demonstrate less fluctuations and let us make a more profound judgment. Another important point is that our data is imbalanced, and, as it was already discussed, certain majority classes mostly contain very short utterances, for which the difference in window size is not relevant. Again, more training instances, especially those that belong to the minority classes, could help alleviate the impact of those frequent short utterances.

As for now we cannot tell which window size is optimal for TRADR, we stay with $w = 10$. This window size is used in all other experiments if not stated otherwise.

4.2.2 Sampling from FrameNet data

Earlier we saw that a classifier trained purely on FrameNet data demonstrated rather poor results when it was tested on the TRADR test set. Based on these results we concluded that we cannot reuse that model to classify TRADR data. However, we can sample from FrameNet some additional instances and retrain the basic PAFIBERT model, originally trained only on TRADR, on the updated training set containing both TRADR and FrameNet examples. Theoretically, this could help achieve better performance and generalization ability, as well as provide more diverse contexts to allow the model to learn differences in meaning of ambiguous targets.

In order to check our assumptions, we perform a series of experiments with sampling from the FrameNet corpus. We sample from a subset of FrameNet containing 21,492 instances of frames that are present in the TRADR data (about 12% of the whole FrameNet corpus). The experiments can be split into two groups. The first group includes training models with different portions of blindly sampled data. The second part involves experiments with informed sampling.

Let us start with the blind sampling. The portions of sampled data range from 10% to 100%, and the instances are chosen randomly, regardless of their classes and how many instances these classes have in TRADR data. Each model is trained using the same 5-fold cross-validation approach with the training set being a mixture of TRADR and sampled FrameNet data, and the validation set coming solely from TRADR data.

The performance of these models on our three TRADR test sets is shown in Table 4.5. We can observe that the IBA scores of most of the models are comparable to or slightly better than the IBAs of the basic model. The best scores (given in bold) are achieved by the model with 40% of sampled data, in this case the improvement makes up 1% for TRADR test sets A and B, and 2% for test set C. However, we cannot say that sampling always demonstrates some positive effect, as three models with sampling presented in the table show slightly worse IBA scores than the basic model.

Based on the results shown in Table 4.5, we can conclude the following. If blind sampling brings any improvement, it is not really significant, namely no more than 1-2% in terms of IBA. There seem to be no strict correlation between the sampled data size and performance, e.g., models with 10%, 40% and 100% of sampled data have almost identical IBA, and a model with 20% of sampled data performs worse than the one with 10%. One possible explanation of such fluctuations in scores may be the stochastic nature of the training and testing procedures and an insufficient test set size. Another possible reason for such results can be the fact that a subset of the FrameNet data devised for sampling only contains a small amount of really useful instances that are not always sampled because of the random character of the sampling procedure. So, instead of sampling from FrameNet blindly, we could focus more on sampling instances of certain classes, especially of the minority ones, and thus try to make our training data more balanced.

This leads us to the second group of the experiments which assume informed sampling. In what follows we will discuss two informed sampling approaches that we devised to overcome the main shortcoming of blind sampling, namely picking out the instances at random regardless of their distribution in both training data and data held out for sampling.

# sampled inst.	Test set	*PRE*	*SEN*	*SPE*	*F*1	*GM*	$IBA_{0.1}$	*SUP*
2,149 (10%)	TRADR set A	0.92	0.90	1.00	0.90	0.94	0.89	268
	TRADR set B	0.91	0.89	1.00	0.89	0.93	0.88	247
	TRADR set C	0.91	0.88	1.00	0.89	0.93	0.87	234
4,298 (20%)	TRADR set A	0.91	0.87	1.00	0.88	0.92	0.86	268
	TRADR set B	0.91	0.87	1.00	0.88	0.92	0.86	247
	TRADR set C	0.91	0.86	1.00	0.87	0.92	0.85	234
6,447 (30%)	TRADR set A	0.91	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.91	0.88	1.00	0.89	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.89	0.93	0.87	234
8,596 (40%)	TRADR set A	0.92	0.90	1.00	0.90	0.94	**0.89**	268
	TRADR set B	0.92	0.89	1.00	0.90	0.93	**0.88**	247
	TRADR set C	0.92	0.89	1.00	0.90	0.93	**0.88**	234
10,746 (50%)	TRADR set A	0.92	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.91	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.87	234
12,895 (60%)	TRADR set A	0.91	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.91	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.89	0.93	0.87	234
15,044 (70%)	TRADR set A	0.92	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.92	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.92	0.88	1.00	0.89	0.93	0.87	234
17,193 (80%)	TRADR set A	0.91	0.88	1.00	0.88	0.93	0.87	268
	TRADR set B	0.90	0.87	1.00	0.88	0.92	0.86	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.87	234
19,342 (90%)	TRADR set A	0.92	0.88	1.00	0.89	0.93	0.87	268
	TRADR set B	0.92	0.87	1.00	0.88	0.92	0.86	247
	TRADR set C	0.91	0.87	1.00	0.88	0.92	0.86	234
21,492 (100%)	TRADR set A	0.91	0.90	1.00	0.90	0.94	0.89	268
	TRADR set B	0.91	0.89	1.00	0.89	0.94	0.88	247
	TRADR set C	0.91	0.88	1.00	0.89	0.93	0.87	234
PAFIBERT trained on TRADR (basic model)	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.5: Performance of models with different percentage of sampled FrameNet data

The first approach, which we call balanced sampling, assumes sampling for each class in the training data a number of FrameNet instances limited by the maximal class population in the TRADR training data. For example, the frame *'Capability'* is the most frequent in our training data - it has 247 inhabitants. This means that the number of FrameNet instances that we want to sample for each class is defined as a difference

between 247 and the current amount of training examples in this class. Considering that the FrameNet corpus is a large one and for many frames it is easy to find the necessary number of additional samples, such an approach makes our training data much more balanced. In total, we sample 10,902 instances (around 51% of FrameNet data held out for sampling). As a result, almost half of all frames in our training data contain more than 200 examples, and only three frames have less than 10 data points.

However, this method also has a potential disadvantage. In case the number of original TRADR utterances is small, and the number of sampled instances is much larger with their targets being different from those in the original utterances, the model will be biased towards dominating training samples and thus prone to misclassification of the TRADR test examples. To avoid a situation, when sampled instances dominate the original ones, we introduce another informed sampling approach with an additional constraint saying that the number of sampled examples cannot exceed the number of the original ones. We call it equal sampling. As an example assume that the frame *'Becoming_aware'* has only 25 instances in the TRADR training data. Now, instead of sampling 222 additional FrameNet examples to reach the limit of 247 samples, we sample only 25 FrameNet sentences. Following this approach, we sample 1,622 instances (about 7.5% of FrameNet data for sampling).

Both models were trained using 5-fold cross-validation strategy. Their performance on all TRADR test sets is given in Table 4.6. The scores show that unfortunately the informed sampling does not perform better than a blind one. Equal sampling produces a little bit higher scores than balanced sampling. These scores are comparable with the scores of the basic model, and are slightly worse, namely by 1%, than those demonstrated by the best model with blind sampling (see Table 4.5). As the difference in scores is really small, it is difficult to conclude, whether it is significant, or is just a side effect of stochastic fluctuations.

Sampling type	Test set	*PRE*	*SEN*	*SPE*	*F*1	*GM*	$IBA_{0.1}$	*SUP*
Balancing 10,902 inst. ($\approx$ 51%)	TRADR set A	0.91	0.88	1.00	0.88	0.93	0.87	268
	TRADR set B	0.91	0.87	1.00	0.88	0.92	0.86	247
	TRADR set C	0.91	0.86	1.00	0.87	0.92	0.85	234
Equal 1,622 inst. ($\approx$ 7.5%)	TRADR set A	0.92	0.89	1.00	0.89	0.93	**0.88**	268
	TRADR set B	0.91	0.88	1.00	0.89	0.93	**0.87**	247
	TRADR set C	0.92	0.88	1.00	0.89	0.93	**0.87**	234
PAFIBERT trained on TRADR (basic model)	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.6: Performance of models with different informed sampling approaches

In order to understand, why neither of the informed sampling approaches brings a desired performance improvement, we compare the errors that these two models make with the errors made by the basic model, and try to analyse the (lack of) influence of the sampled data. We use the TRADR test set A for this purpose. The errors grouped by frames are presented in Table 4.7. The table includes only those frames from the

test set, whose instances were misclassified by the basic model. The frames are sorted by the number of errors in descending order (the fifth column). The third column shows the number of frame instances in the training data before sampling. The next column demonstrates the number of samples of these frames in TRADR test set A. The rest of the table presents the number of sampled examples for each frame, and shows how the number of errors changed depending on the sampling approach. Green is used to mark a positive influence of sampling, pink - a negative one.

According to Table 4.7, we can say that sampling has a positive impact only on a few frames. All of these frames have less than 30 instances in the original training set. However, the number of added examples seems to have no influence in these cases, i.e. it does not matter if we sample 10 or 100 additional sentences. In case of the *'Arriving'* frame, adding more samples seems to have a positive effect, but this may be just a chance result, similar to the case with the *'Physical_entity'* frame, where no additional examples were sampled at all. Sampling may lead to contradictory results, as in case of the *'Possibility'* frame. We can also notice that sampling seems to have a negative effect on recognizing *'Scrutiny'* and *'Perception_active'* frames. It should also be noted that both sampling types lead to a couple of more errors that are not included in Table 4.7, as these instances were classified correctly by the basic model. In total, balanced sampling resulted in six additional misclassification cases, and equal sampling - in three such cases. Finally, we also need to mention that more than half of the frames in Table 4.7 experienced absolutely no effect from any sampling type.

#	Frame	# occ. in TRADR train	# occ. in TRADR test	# incorr. classified	Balanced sampling		Equal sampling	
					# sampled	# incorr. classified	# sampled	# incorr. classified
1	Being_obligated	28	6	3	37	1	28	2
2	Motion	177	21	2	70	2	70	2
3	Attempt	33	13	2	214	2	33	2
4	Create_representation	103	12	2	45	2	45	2
5	Perception_experience	169	25	2	78	2	78	2
6	Attempt_suasion	16	3	2	231	1	16	1
7	Inspecting	74	6	2	173	2	74	2
8	Presence	55	3	2	99	2	55	2
9	Event	6	1	1	236	1	6	1
10	Desirable_event	12	1	1	74	0	12	0
11	Arriving	27	2	1	220	0	27	1
12	Traversing	12	2	1	235	1	12	1
13	Becoming_aware	25	1	1	222	0	25	0
14	Physical_entity	9	3	1	0	0	0	1
15	Holding_off_on	6	3	1	13	1	6	1
16	intentionally_act	10	2	1	237	1	10	1
17	Scrutiny	60	4	1	187	2	60	2
18	Perception_active	35	5	1	212	2	35	2
19	Cause_to_move_in_place	7	1	1	80	1	7	1
20	Possibility	19	5	1	62	3	19	0
21	Comm._response_message	79	7	1	0	1	0	1

Table 4.7: Basic model vs. models with informed FrameNet sampling: error comparison

If we have a look at the actual utterances from the TRADR test set A that were misclassified, we can find some explanations for such an unsatisfactory sampling effect. In total, the basic model makes 30 mistakes. We can divide them into four main groups.

The first group is the largest and includes 22 mistakes that were made because of ambiguous targets, such as *'do'*, *'have'*, *'try'*, *'turn'*, etc. in various forms. The fact that in most cases neither the target, nor its context give any clues what the true frame may be, makes the situation even more complicated. For instance, the basic model identifies the utterance *"I've already done that"* with *'done'* marked as target as an instance of

the *'Intentionally_ act'* frame, while the true frame is actually *'Create_ representation'*. Or the utterance *"... you have to take a close look"* with the target *'have to'* is labeled as *'Imposing_ obligation'* with the true frame being *'Being_ obligated'*. In many of these cases a predicted frame and a true one can be actually rather close semantically, or several interpretations can be possible given a certain target. For example, *'Capability'* (predicted) and *'Attempt_ suasion'* (true) frames for the utterance *"... but you can take a look at the picture box"* with the target *'can'*, or *'Inspecting'* (predicted) and *'Perception_ active'* (true) for *"Can you go to the victim and look if he is...?"* with the target *'look'*, or *'Capability'* (predicted) and *'Possibility'* (true) for *"I could circle around the barrel and find out if there' s a label on it"* with the target *'could'*. And we see that unfortunately sampling more examples of such frames does not help resolve the ambiguity issue. So, e.g., equal sampling contributes to the correct classification of three cases out of 22, but results in two new errors also caused by ambiguity.

The second group contains two errors caused by incorrect targets that in their turn, come from incorrect reading of some wrongly formatted lines in the input file. Apparently, the basic model fails to assign correct frames, because these targets contain several tokens, which make them confusing. Both sampling approaches cannot solve this issue.

Besides all this, we also have one misclassification case that happens because of an inaccurate translation from German into English, when the verb *'discover'* was used instead of *'explore'* and *'inspect'* that normally evoke *'Inspecting'* or *'Scrutiny'* frames. As a result, the basic model assigns the frame *'Scrutiny'* instead of *'Becoming_ aware'* to the utterance *"So just to discover what I see"* with the target *'discover'*. In this case sampling has a positive effect, and the utterance is classified correctly.

Finally, there are five more errors that do not belong to any of the three previous groups. These are silly mistakes that cannot be attributed to the similarity between frames, ambiguous targets and so on. For instance, the utterance *"Which robot is now assigned to casualty search?"* with the target *'assigned'* got the label *'Create_ representation'* instead of *'Being_ obligated'*. The effect of sampling on such cases is also controversial. On the one hand it helps eliminate three errors, on the other hand two new silly mistakes are made.

All in all, we have to conclude that data augmentation with sampling from FrameNet turned out to be unable to bring any stable performance gain. However, it is possible that sampling only certain frames, like those marked with green in Table 4.7, would be more effective. Such an approach requires more investigation and evaluation of the models on various test sets in order to pick out the most useful frames. Also, it can happen that sampling could still be beneficial in cases when the model is evaluated on different data from the same or similar domain. All this can be a topic for future research.

4.2.3 Adding features

Our next attempt to improve the performance of the PAFIBERT classifier trained on TRADR assumes introducing additional features for input tokens and sequences as a whole. According to Sundararaman et al. [2019], several studies showed that, e.g., in the machine translation area adding to the model such features as part of speech (POS) and

named entity tags provides a slight improvement over baseline for small size training datasets. Sundararaman et al. [2019] perform their own experiments, in which they modify the input embeddings with syntactic features, namely POS tags, grammatical cases and subword masks. They motivate their choice of features by the fact that POS tags and cases may help distinguish between important and common words, and subword masks can bring cohesion in cases when tokens get split by the BERT tokenizer. The authors report that their models outperform $\text{BERT}_{\text{BASE}}$ model on four General Language Understanding Evaluation (GLUE) [Wang et al., 2019] benchmarks. They call their approach a syntax-infused transformer.

Following Sundararaman et al. [2019], we decided to check if introducing additional features would have any positive impact on the performance of our model. We divide them into two groups: lexical features that include POS tags and subword masks, and discourse features that contain speaker tags and dialogue acts.

Let us first discuss the lexical features and their influence on the basic model. In contrast to Sundararaman et al. [2019], we call this group 'lexical', because we only have POS tags and subword masks here, and they do not capture any relations between tokens, word order, etc. We deem these features useful for the following reasons.

First of all, we have cases, when the POS tag of a target may be important to differentiate one frame from another. For example, in the utterance *"Can you position yourself onto the track?"* the target *'position'* is a verb and evokes the *'Placing'* frame, while in the utterance *"What's your current position?"* *'position'* is a noun that induces the frame *'Locale_by_collocation'*. A POS tag feature can serve for frame disambiguation purposes here.

Second, because BERT tokenization splits the tokens that are not included into the tokenizer vocabulary, sometimes it happens that some parts of a token are not included into the target context. The utterance *"If you can make some pictures , u ##g ##v two , then it give me some v - overview"* is an example of such a situation. Given the target *'overview'* and the context window of ten tokens, we can notice that two parts of the token 'UGV' are not part of the context. In this very example this fact may be not relevant, however, it may be important in other cases, e.g., when a split token is a dependent of the target and can help disambiguate it.

So, to tackle such issues, we provide our basic model with POS tags of the input tokens, as well as the information about whether these tokens were split by the BERT tokenizer or not. In order to perform the POS tagging we use a tagger from the *Python SpaCy* library [Honnibal and Montani, 2017]. There are 19 coarse-grained tags that follow the Universal Dependencies scheme. They do not code any morphological features, such as tense, aspect, degree and so on, and only cover the word type. We add two more tags to this set. First, BERT expects each input sequence to be extended with two special tokens, namely [CLS] and [SEP] to mark the beginning and the end of the sequence, respectively. We introduce a tag SPECIAL to mark these special tokens and separate them from 'normal' ones. Next, we add the tag PAD for padded tokens, as BERT demands that all sequences should be of the same length. If a token gets split by the tokenizer, each sub-token is assigned the POS tag of the original word.

As for the subword masks, our implementation is a little bit different from the one described by Sundararaman et al. [2019]. They use IOB format to label tokens and their parts, i.e. whole tokens get the tag O, those that are split get the tag depending on

the sub-token position. So, the first sub-token is labeled with the tag B ('beginning'), the ones in the middle - with the tag I ('inside'), and the last sub-token - with the tag E ('end'). Our approach is simpler. The mask is set to a positive bit (1), if the given element is a sub-token, and to a negative bit (0), if the token is intact.

It should be mentioned, that at first in order to train a model with additional feature(s), we tried to follow one of the two methods suggested by Sundararaman et al. [2019]. Both imply training the embeddings for the additional features along with the model itself. According to the first method, the embeddings for the additional features have the same dimensions as the $\text{BERT}_{\text{BASE}}$ hidden layer $h = 768$, and are summed up with the input embeddings before being fed to the transformer blocks. The second method assumes the concatenation of the embeddings and their affine transformation to BERT hidden size of 768 before feeding the embeddings to the transformer blocks. In this case additional embeddings can be of any length. As Sundararaman et al. [2019] note, the second option is more robust, but it requires learning a large matrix to perform the affine transformation, which is problematic for tasks with little training data. As the authors did not actually implement the second approach, we decided to go with the first one. After feeding our summed up embeddings to the $\text{BERT}_{\text{BASE}}$ model, we trained the rest of the PAFIBERT model as usual (see Section 4.1.1). Unfortunately, this did not bring us any performance gain. On the contrary, after the 5-fold cross-validation procedure the averaged validation IBA score for the five folds was only about 84%, which is worse than the IBA result of the basic model. The learned model was not evaluated on the TRADR test sets. A possible explanation of the unsatisfactory performance of this approach can be the fact that we have much less training data, namely only 2,444 examples, while Sundararaman et al. [2019] train their models using 4.5M instances.

As the syntax-infused transformer was not successful for our data, we tried to incorporate additional features into our baseline model at a later step in the network. As before, we trained embeddings for our feature(s) together with the model, but instead of summing them up with input embeddings and feeding to the transformer blocks, we concatenated them with the $\text{BERT}_{\text{BASE}}$ model output, namely with (sub)token vectors, and used them as input for the position-based attention layer of the PAFIBERT model. Next, instead of introducing one more linear layer to perform the affine transformation to get back to the original hidden layer size, we simply increased the size of the first linear layer of PAFIBERT depending on the dimension of the (sub)token embeddings.

The results of this approach turned out to be better than those of our version of the syntax-infused transformer, but still not good enough to beat the basic model. They are presented in the first three rows of Table 4.8. We test the features separately and in combination. We see that taken separately, the features do not bring any improvement, and the scores are actually slightly worse than the corresponding scores of the basic classifier. The combination of POS tags and subword masks with the tokens representations seems to be more successful - it increases the performance by 1%. Unfortunately, we cannot call 1% increase significant, and it is difficult to judge, whether it was really caused by the combination of features, or is a chance result.

The second group of additional features includes discourse features, namely the speaker tag and dialogue act type. We decided to check these features, because analysing the mistakes made by the basic classifier, we noticed that sometimes in-

formation about who is talking and why could be useful for frame disambiguation. Especially, taken into consideration that utterances can be elliptical and ungrammatical. For instance, given a short utterance *"Try it"* with the target *'try'*, our classifier has difficulties labeling it, because in order to assign the correct frame it needs to know the perspective, i.e. the speaker. If the speaker is the team leader, then the correct frame is *'Attempt_suasion'*, if it is an operator, then it should be the *'Attempt'* frame.

Following Anikina and Kruijff-Korbayová [2019], we use three labels to encode a variety of speakers: MC for the mission commander, TL for the team leader and OPERATOR for all UGV and UAV operators.

The information about the dialogue act type may also be helpful to disambiguate frames. It can be used to strengthen the impact of the speaker tag, because there exist a strong correlation between the speaker and the dialogue act [Anikina and Kruijff-Korbayová, 2019], i.e. certain dialogue acts are typical for certain speakers, e.g., a *'Request'* is normally made by the team leader, and an *'Answer'* is usually given by the operator. If we take the last example, it can be beneficial to know whether *"Try it"* is a *'Request'* or an *'Inform'* dialogue act, if we want to figure out the perspective. Another example is the utterance *"Do you copy me?"* with the target *'copy'*, which is misclassified by the baseline as the *'Communication_response_message'*, when in reality it belongs to the frame *'Perception_experience'*. Providing to the classifier the information that the utterance is actually a question could help exclude *'Communication_response_message'* from the list of candidate frames, as this frame encompasses short answers and thus is incompatible with the dialogue act *'Question'*.

In order to tag each utterance with a dialogue act, we use a simplified set of 12 different labels. Originally, TRADR dialogues were annotated with more than 60 dialogue acts belonging to seven dimensions. We use simplified tags for the following reasons. First, many of the original dialogue acts are rare, they have too few instances for the classifier to learn them. Second, the difference between certain dialogue acts, e.g., between *'Set Question'* and *'Choice Question'* is not really relevant for the task of frame recognition, and such dialogue acts can be merged. We borrow eight simplified tags from the dimensions *'Task'*, *'Communication Feedback'* and *'Turn Management'* from Anikina and Kruijff-Korbayová [2019]. These tags are *'Affirmative'*, *'Confirm'*, *'Contact'*, *'Disconfirm'*, *'Inform'*, *'Negative'*, *'Question'* and *'Request'*. We introduce four more simplified labels to group the dialogue acts that belong to the rest of dimensions, namely *'Communication Management'*, *'Time Management'*, *'Discourse Structuring'* and *'Social Obligations'*. We use the dimensions' names as simplified labels.

Like the embeddings for lexical features, the embeddings for discourse features are trained jointly with the model. However, as they characterize the whole utterance and not separate (sub)tokens, we concatenate them not with BERT input representations, but with the output of the PAFIBERT position-based attention layer, i.e. with the target and context representation for each utterance. We increase the size of the first linear layer in the model accordingly.

Again, we test the discourse features separately and in combination. The results are shown in Table 4.8. In contrast to our expectations, discourse features did not turn out to be really beneficial. We see that the models with a single 'speaker' feature and with both discourse features perform similarly to the basic classifier, and the model with the feature 'dialogue act' demonstrates slightly worse results. We also tried to

combine lexical features with the 'speaker' feature (see the last row of Table 4.8), but this model could not bring any performance gain either.

Feature	test set	*PRE*	*SEN*	*SPE*	*F*1	*GM*	$IBA_{0.1}$	*SUP*
POS tag	TRADR set A	0.89	0.88	1.00	0.88	0.92	0.87	268
	TRADR set B	0.89	0.87	1.00	0.87	0.92	0.86	247
	TRADR set C	0.89	0.87	1.00	0.87	0.91	0.86	234
Subword mask	TRADR set A	0.89	0.88	1.00	0.87	0.92	0.87	268
	TRADR set B	0.89	0.87	1.00	0.87	0.92	0.86	247
	TRADR set C	0.89	0.86	1.00	0.87	0.91	0.85	234
POS tag + Subword mask	TRADR set A	0.91	0.90	1.00	0.90	0.94	**0.89**	268
	TRADR set B	0.90	0.89	1.00	0.89	0.93	**0.88**	247
	TRADR set C	0.91	0.88	1.00	0.89	0.93	**0.87**	234
Speaker	TRADR set A	0.89	0.88	1.00	0.88	0.92	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.92	0.87	247
	TRADR set C	0.90	0.88	1.00	0.88	0.91	0.87	234
Dialogue act	TRADR set A	0.89	0.87	1.00	0.87	0.92	0.86	268
	TRADR set B	0.88	0.86	0.99	0.86	0.91	0.85	247
	TRADR set C	0.88	0.85	0.99	0.86	0.91	0.84	234
Speaker + Dialogue act	TRADR set A	0.90	0.88	1.00	0.88	0.92	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.92	0.87	247
	TRADR set C	0.89	0.87	1.00	0.87	0.91	0.86	234
POS tag + Subword mask + Speaker	TRADR set A	0.88	0.88	1.00	0.87	0.92	0.87	268
	TRADR set B	0.87	0.87	1.00	0.86	0.91	0.86	247
	TRADR set C	0.88	0.86	0.99	0.86	0.91	0.85	234
PAFIBERT trained on TRADR (basic model)	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.8: Performance of models with different additional features

It is interesting to mention that if we have a look at the mistakes that the classifiers with discourse features make, we can see that all these models are not able to distinguish between different perspectives despite the fact that we provide them with information about the speakers. For instance, the models still have difficulties differentiating between the *'Attempt'* and *'Attempt_suasion'* frames, which have a strict correlation with the speakers: in our training data 27 out of 33 instances of the frame *'Attempt'* have OPERATOR as speaker, and 14 out of 16 instances of *'Attempt_suasion'* frame have TL as speaker. Information about dialogue acts seems to be useless either, e.g., the utterance *"Do you copy me?"* with the target *'copy'* is still labeled as *'Communication_response_message'* despite the fact that we inform the classifier that it is a question.

It is difficult to say why both lexical and discourse features do not bring us any significant performance improvement. One of possible hypotheses is that our learned

feature embeddings are rather short (2-4 neurons) in comparison with input embeddings (768 neurons) or context-target embeddings (1536 neurons), so their impact on the whole (sub)token/utterance representations is actually negligible. We think that in order to get a better estimation of the role of additional features, some further experiments with more data are necessary.

4.2.4 Frame filtering

Analysing the mistakes made by the frame classifiers, we noticed that some mistakes are rather silly and cannot be explained by target ambiguity or by semantic closeness between predicted and true frames. For example, given the utterance *"I put my robot on hold and UGV-2 get a new battery"* and a target *'put on hold'*, our basic classifier assigns *'Create_representation'* frame to it, while the real frame is *'Holding_off_on'*. The reason for such sorts of errors is the fact that while learning the relation between a target and its context and the true frame, we assume by default all available frames (there are 81 of them) to be equally likely. So, in order to perform classification, our *Softmax* function has to process all candidates, and in some cases, e.g., if we do not have enough training samples, it can happen hat several candidate frames (often not related) have equal chances to be predicted as the true frame, and the classifier makes its choice completely at random.

To avoid such mistakes, we can try to inform our classifier about candidate frames and make it focus only on those that make sense. In other words, we can apply frame filtering. In a way, it is similar to weighting the data space, which is used to deal with imbalanced data (see Section 2.4). Data space weighting assumes giving training examples certain weights to increase/decrease their importance depending on their classes, and frame filtering does the similar thing with candidate frames - frames that are never evoked by the given target are given zero weight and thus are excluded from the list of candidates.

Frame filtering is proposed as a variant of the PAFIBERT model by Tan and Na [2019]. The authors just add an additional frame filtering layer to the $\text{PAFIBERT}_{\text{BASE}}$ model as shown in Figure 4.3. So, the output of the position-based attention layer goes through the first linear layer and a *Tanh* activation function, and the outcome is fed into the second linear layer W_k of size $k \times d$, where k is the number of classes. Next, instead of being fed directly to *Softmax*, the result is multiplied element-wise with a so-called candidate frame vector, and is being further processed by the third linear layer of size $k \times k$ and the *Softmax* function.

So, the candidate frame vector can be viewed as vector that contains weights for all possible frames in the model depending on target. If the weights are set to ones, the model will behave similar to $\text{PAFIBERT}_{\text{BASE}}$. Setting certain frames to zero values, we exclude them from candidates list. If the target has only one sense, then only one candidate frame is present. If the target is ambiguous, then the search space is limited by several candidate frames. Of course, for this approach to work really well we need an exhaustive list of targets and all their senses, which is impossible. So, if we come across an unseen target in our test data, the model will have to consider all possible frames.

Tan and Na [2019] construct candidate frame vectors using two methods. The first

assumes frame filtering by lexical units of the targets, the second one - frame filtering by targets. Both imply creation of look-up tables. According to the first method, they map each lexical unit found in FrameNet to a range of frames that this unit evokes, which are also retrieved from the FrameNet database. However, if test data comes from a source other than FrameNet, the lexical units are usually unknown. The second methods was introduced to tackle such a situation. It suggests an additional mapping of targets to lexical units. In order to create this mapping, Tan and Na [2019] utilize the *Pattern* library for *Python* [De Smedt and Daelemans, 2012]. They use it to get various word forms for all lexical units from the FrameNet database. These word forms are meant to cover possible targets, and are mapped to the corresponding lexical units. Depending on the method, Tan and Na [2019] report the accuracy improvement by 1-2.5%, when filtering is applied.

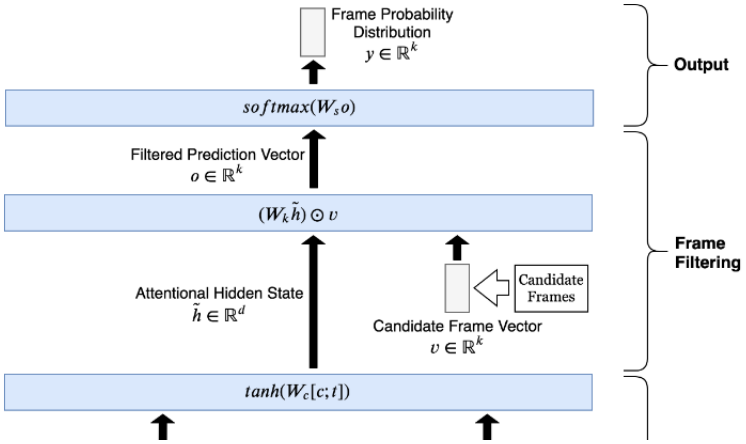


Figure 4.3: PAFIBERT: a filtering layer (taken from Tan and Na [2019])

We follow the first method suggested by Tan and Na [2019], and create candidate frame vectors using filtering by lexical units. Our look-up table is based on the training data. This means that given a test example, the corresponding candidate frame vector, that we retrieve for the specified target, contains only those frames that were observed in the training set with respect to this target.

Our original implementation of frame filtering layer also follows the one presented by Tan and Na [2019] and depicted in Figure 4.3. Because the classifier with two lexical features demonstrated the best IBA scores, we chose it for the experiments with filtering instead of the usual basic model. So, the classifier got an additional filtering layer and was trained using the 5-fold cross-validation procedure with the usual eight epochs per fold. However, already the averaged validation IBA score after the 5-fold cross-validation procedure turned out to be very low - only about 37%. When we increased the number of epochs up to 16, the IBA score grew up to about 53%. Because of the clearly unsatisfactory performance, we did not evaluate this model on the test sets. We attribute such low validation accuracy to the fact that we have little training data, but need to train an additional weight matrix of size $k \times k$, where k stands for the number of classes.

Next, we tried a slightly different filtering approach. We excluded the very last hidden layer W_s and applied the *Softmax* function directly to the output of the frame

filtering layer, namely to the vector $o \in \mathbb{R}^k$ (see Figure 4.3). This modification allowed to boost the model's performance, so that the validation IBA score grew up to 93% (average value for five folds).

This model was evaluated on our usual TRADR test sets. The results are given in Table 4.9. One can notice that the model with filtering still was not able to make it past 88% IBA, which seems to be the upper bound.

Model	test set	PRE	SEN	SPE	$F1$	GM	$IBA_{0.1}$	SUP
POS tag + Subword mask + Filtering	TRADR set A	0.90	0.88	1.00	0.88	0.92	0.88	268
	TRADR set B	0.89	0.87	1.00	0.87	0.92	0.86	247
	TRADR set C	0.90	0.88	1.00	0.88	0.92	0.87	234
PAFIBERT trained on TRADR (basic model)	TRADR set A	0.90	0.89	1.00	0.89	0.93	0.88	268
	TRADR set B	0.90	0.88	1.00	0.88	0.93	0.87	247
	TRADR set C	0.91	0.88	1.00	0.88	0.92	0.86	234

Table 4.9: Performance of the model with frame filtering mask

In order to understand the impact of filtering (or a lack of it) we perform careful analysis of the mistakes made by the classifier. It leads us to the following conclusions. Most of the mistakes, namely 19 out of 31 are caused by ambiguous targets. This is the most challenging type of errors, and we suppose that they should be considered the main problem that needs further consideration. Given the same test data, their proportion among other types of mistakes usually makes up approximately 60 - 70% regardless of the model. For now, such fluctuations seem to be a result of stochastic nature of the training process.

Filtering allows to avoid errors coming from incorrectly parsed targets as long as the corresponding lexical units are read correctly. It also helps understand why certain types of errors are made. For example, earlier we mentioned the so-called silly mistakes, when the predicted frames semantically have nothing to do with targets and context, like in case of the utterance *"I put my robot on hold and UGV-2 get a new battery"* with the target *'put on hold'*, which was labeled as *'Create_representation'*. Now we know that such errors are caused by the fact that some targets/lexical units that occur in the test data are absent in the training data, and as a result the model cannot focus on any particular candidates and chooses the class to large extent at random. Among the mistakes made by our classifier with filtering, 7 out 31 cases happen for this reason.

One more group of mistakes includes cases when an utterance is being misclassified, because its target was used in a new sense that is not present in the look-up table. This fact explains the last 5 mistakes out of 31 made by our classifier with filtering. For instance, the utterance *"Yes, that was quite a while ago"* with the target *'was'* was incorrectly labeled as *'Identity'*, while the true frame was *'Event'*. This happened, because based on the training data look-up table for the lexical unit *'be.v'* only contains such frames as *'Identity'*, *'Being_located'*, *'Presence'* and *'State_of_entity'*. The frame *'Event'* is not on this list, and only occurs in the test data.

All in all, we see that in its nature filtering is a helpful feature. However, it works only if sufficient training data is available. The last two groups of mistakes made by

the classifier with filtering could have been avoided if the look-up table with frame candidates had a better coverage.

4.3 German frame classifier

In this section we will present the German version of the semantic frame classifier for TRADR data. We define the German basic model similarly to the English one. It is the same $\text{PAFIBERT}_{\text{BASE}}$ model described earlier but trained exclusively on German TRADR data. Again, we use a context window $w = 10$, the contextualized token embeddings are represented by the last hidden layer of the $\text{BERT}_{\text{BASE}}$ model, and no sampling or extra features are applied. The model is trained with 5-fold cross-validation, and we save the best model out of five. Each fold is trained for eight epochs with an adaptive learning rate which starts with 3e-5.

The basic model's performance on both German test sets is shown in Table 4.10. We see that the IBA score reaches 81% - 83%, which is about 5% less than the IBA score of the analogous basic model trained on the English TRADR dialogues. The reason for this difference can be the fact that the English training data is larger - in total it contains 2,444 utterances, while the German training data consists of 2,378. If we consider that on average a TRADR utterance in English has 8.68 tokens, and a German one - 6.99 (see Table 3.3), we can easily calculate that the training data in English totals about 21,214 tokens against approximately 16,622 ones in the German training data.

Now let us have a look at the mistakes that the basic classifier makes. In what follows we will discuss the results given the test set A that contains 259 instances. Here the basic classifier makes incorrect frame assignments for 42 utterances, and the errors can be divided into three groups.

The first group includes 26 instances with ambiguous target related elements. Among them we have 18 cases where the target related elements are actually represented by a single token or by a reflexive verb, e.g., the frame classifier assigns a frame *'Posture'* instead of *'Presence'* to the utterance *"Bin auf dem Treppenabsatz aber weiter links gelegen"* with the target *'gelegen'*, or a frame *'melden2-salsa'* instead of *'Contacting'* to the utterance *"Wir sind unterwegs und melden uns gleich nochmal"* with the target *'melden uns'*. Similar to the English data, most of these instances do not contain any clues to help perform disambiguation. Especially ambiguous are the targets represented by various forms of such verbs as *'sein'*, *'haben'*, *'mögen*, *'müssen'*, *'gehen'*, etc. Further 8 cases in this group include utterances where the target expression consists of several tokens. It can be a stable expression or a verb with a prepositional adverb, e.g., *'vorbei gefahren'* in the utterance *"... und die UGV 2 müsste da dran vorbei gefahren sein"*, which got the label *'Motion'*, while the correct one was *'Traversing'*. Or just the whole syntactic phrase, e.g., *'draußen steht'* in the utterance *"Draußen steht ein Löschzug mit Blaulicht"* which was classified as *'Presence'* instead of *'Locative_relation'*. Judging by this examples, we can conclude that multi-token targets are not perceived by the classifier as a whole, it tends to focus on a single word, and such multi-token target expressions are actually confusing.

The second group encompasses 14 cases, which we define as simple mistakes. The

target related elements here may consist of one or more tokens, but none of these tokens normally evokes the frame that was assigned by the classifier. An example of such a situation is the utterance *"Schwenk mal nach rechts, dann müsstet ihr eigentlich in die Halle reinblicken"* with the target *'reinblicken'* which was misclassified as *'Motion'* with the right frame being *'Perception_active'*. It is interesting to note that sometimes the label assigned by the classifier is considered to be wrong, but is still plausible given the context. E.g., *"... sag mir, wo du hin möchtest"* with the target expression *'hin möchtest'* was classified as *'Motion'*, while the correct frame is *'Desiring-fnsalsa'*. Another example would be the sentence *"Aber teilweise unfähig das umzusetzen, weil mein Roboter sich teilweise gar nicht bewegt"* with the frame related elements *'sich gar nicht bewegt'* which got the label *'ausfallen1-salsa'* instead of *'Motion'*. In both cases the gold frame corresponds to the direct meaning of the target expression, while the assigned one assumes the interpretation of the expression in the context.

The last group is rather small and includes two misclassified utterances. Both utterances have incorrect targets, which can happen due to annotation or processing errors. Normally, such cases are rare. So, the utterance *"Also - die Person müsste hinter dem großen Hochofen auf dem freien Weg sein"* got the label *'Presence'* instead of *'Locative_relation'*, because the target expression *'sein'* was wrong. Note that in principle it is possible for *'sein'* to evoke the *'Presence'* frame. Similarly, the utterance *"Ich würde jetzt einfach geradeaus fahren und schauen was ich so weiter finde. Kommen"* was recognized as an instance of the *'Communication_response_massage'*, while the correct frame was *'Communication_by_protocol'*. This happened, because *'Ich'* was incorrectly marked as the target expression instead of *'kommen'* that normally evokes the *'Communication_by_protocol'* frame.

Model	test set	*PRE*	*SEN*	*SPE*	*F*1	*GM*	$IBA_{0.1}$	*SUP*
Basic model	TRADR set A	0.84	0.84	0.99	0.83	0.89	0.83	259
	TRADR set B	0.84	0.82	0.99	0.82	0.88	0.81	217
Basic model + balanced sampling	TRADR set A	0.89	0.89	1.00	0.88	0.93	0.88	259
	TRADR set B	0.90	0.88	1.00	0.88	0.92	0.87	217
Basic model + equal sampling	TRADR set A	0.91	0.91	1.00	0.90	0.94	**0.90**	259
	TRADR set B	0.91	0.90	1.00	0.90	0.93	**0.89**	217
Basic model + POS tag	TRADR set A	0.87	0.86	0.99	0.86	0.91	0.85	259
	TRADR set B	0.88	0.86	0.99	0.86	0.91	0.85	217
Basic model + subword mask	TRADR set A	0.86	0.84	0.99	0.84	0.90	0.83	259
	TRADR set B	0.86	0.83	0.99	0.83	0.89	0.82	217
Basic model + POS tag + subword mask	TRADR set A	0.84	0.84	0.99	0.83	0.89	0.82	259
	TRADR set B	0.84	0.84	0.99	0.83	0.88	0.83	217
Basic model + POS tag + equal sampling	TRADR set A	0.87	0.85	1.00	0.85	0.90	0.84	259
	TRADR set B	0.87	0.85	0.99	0.85	0.90	0.84	217

Table 4.10: German frame classifier

Like in case with the basic model trained on English TRADR data, we also investigate the influence of sampling on the performance of the basic German frame classifier. As

earlier, we apply two types of sampling, namely balancing and equal. As a source of sampling we use SALSA data. Because we sample only the instances of those frames that occur in German TRADR data, most of the samples are filtered out from the SALSA corpus, so that only 2,486 out of 35,236 are left.

In case of balanced sampling we sample 2,375 instances from 2,486 available ones, and retrain the classifier without changing any parameters. As Table 4.10 demonstrates, this type of sampling allows to increase the IBA score up to 87% - 88%, which is about 5% more than the score reached by the basic model. If we have a look at the mistakes made by the classifier with balanced sampling, we can notice the following. The number of simple errors is considerably reduced, namely from 14 to 8 cases. Next, the number of mistakes caused by ambiguous target expressions also gets smaller - we have 19 incorrectly classified instances against 26 ones when testing the basic model. The last reduction is mostly determined by a better recognition of ambiguous target expressions consisting of a single token. So, while the basic classifier misclassifies 18 utterances with such targets, the classifier relying on balanced sampling - 14. Finally, we must mention that balanced sampling does not influence the two mistakes caused by the incorrect data annotation/preprocessing.

For the second sampling approach - equal sampling - we randomly pick out 611 instances from 2,486 SALSA sentences left for sampling. This allows the model to achieve an even better IBA score of 89% - 90%. Again, the increase of IBA is explained by the fact that the amount of simple errors, as well as errors determined by ambiguous single-token targets, decreases. However, because of the small test set size, it is difficult to tell if the difference in performance of the models with balanced and equal types of sampling is really significant. So, the classifier with the equal sampling in total makes only 5 mistakes less than the classifier with the balanced one. Considering that we deal with ambiguous targets, there always exists a possibility to assign the correct frame by chance. E.g., our test set A contains two instances of the *'Grasp'* frame, both of which have the target *'verstanden'* that usually evokes either *'Grasp'* or *'Communication_response_message'* frames. So, while the classifier with balanced sampling misclassifies both instances as *'Communication_response_message'*, the classifier with equal sampling misclassifies in a similar way only one of them. As the SALSA corpus does not contain any samples of the latter frame at all, we cannot say that the equal sampling allows the classifier to learn the difference between *'Grasp'* and *'Communication_response_message'* better than the balancing one. It is clear that in case with the equal sampling one of two instances was classified correctly by pure chance.

Speaking of sampling, it is necessary to mention another interesting detail, namely the fact that despite showing different IBA results when evaluated on test data (see Table 4.10), the basic model and the model with equal sampling (we omit the balanced model from discussion as it behaves likewise) have the same averaged validation IBA scores, as the second and the third columns of Table 4.11 demonstrate. The table contains the validation scores of all of the five folds for each model, the best ones are given in bold.One of the possible hypotheses of why the two models have different test IBA scores (about 83% and 90%) but the same averaged validation ones can be the usage of dropout during the 5-fold cross-validation procedure. Due to dropout some neurons in the network get 'switched off' and in case of the model with sampling some useful information coming from the additional SALSA instances may get lost as a

result. However, during the testing the model has all units available, and this allows it to achieve better scores on the test sets. In addition to this, the small sizes of the test sets often result in situations when even a small amount of chance correct assignments inflates the IBA scores.

Model	Basic model	Basic model + equal sampling	Basic model + POS tag	Basic model + POS tag + equal sampling
Fold 1	0.83	0.84	0.83	0.84
Fold 2	0.84	0.84	**0.85**	0.83
Fold 3	**0.86**	**0.85**	0.84	**0.86**
Fold 4	0.81	0.82	0.80	0.82
Fold 5	0.84	**0.85**	0.83	0.81
Avg.	0.84	0.84	0.83	0.83

Table 4.11: Validation IBA scores for different folds during cross-validation for some of the models

We also need to point out one more observation regarding sampling, namely the influence of sampling on the classifier performance depending on language. So, in case with the classifier trained on the English data, sampling did not seem to work at all. No matter if we applied direct sampling of certain percentage of FrameNet data, or informed sampling, the IBA score stayed approximately the same. However, we can observe the positive effect of sampling for the classifier trained on the German data - equal sampling actually helped to correct more than half of the mistakes made by the classifier that was trained without sampling. Our hypothesis about why it is so is as follows. It looks like the size of the training data plays the crucial role here. We know that the English training data contains almost 1.3 times more tokens than the German one. Apparently, sampling can help if the training set size is small enough, but at the same time the upper bound in terms of the training set size is rather low, i.e. if we had more German training data, the effect from sampling would probably be comparable with that of for the English training data. In addition to this, the differences between languages (i.e. in morphology, syntax, semantics) may also be important. E.g., the basic frame classifier trained on the English TRADR data misclassified 30 out of 268 instances, and the absolute majority of errors, namely 22 (more than 73%), were caused by ambiguous targets, and only 5 errors (less than 17%) were simple misclassifications. In contrast to this, the percentage of simple errors made by the German frame classifier was more than 33% (14 out of 42 cases), and the fraction of misclassified utterances with ambiguous targets was smaller - about 62% (26 out of 42 errors). So, judging by this, we can conclude that sampling probably helps to resolve simple mistakes, but it is much less effective in cases where disambiguation is necessary. And this can be another reason why it worked for the German frame classifier but failed for the English one.

The last row of experiments with the frame classifier for German TRADR data includes several modifications of the basic model which assume extending token embeddings with lexical features. The modifications correspond to those performed on the English model. We use the same lexical features: POS tags and subword masks.

We omit the discourse features, namely speaker and dialogue type, because despite the fact that all German dialogues were previously annotated with speakers and dialogue acts, the currently used data format does not include this information. The results of the experiments are also given in Table 4.10.

First, we investigate the influence of lexical features without sampling any additional instances from the SALSA corpus. It is interesting that extending token embeddings with the corresponding POS tag embeddings seems to have a positive effect on the IBA score - in comparison with the score of the basic model it is about 3%-4% higher. In order to check where exactly the improvement is, we had a look at the mistakes made by the frame classifier.

We found out that in total the classifier incorporating the information about POS tags makes 36 mistakes when tested on the test set A. This is 6 mistakes less than the basic classifier made. If we compare the number of mistakes in each of the three groups, we can notice that the modified classifier seems to be able to deal with ambiguous target expressions better than the basic one. Actually, the ambiguous targets is the only group out of three, where the number of errors decreased. E.g., the classifier correctly identified two utterances with targets *'liegen'* and *'liegt'* as instances of the *'Posture'* frame instead of *'Presence'*, as well as two utterances both with the target *'ist'* as instances of the frame *'Presence'* in place of *'Identity'*. However, it is unclear how exactly the information about the POS tags contributed to the correct frame assignment in the above mentioned cases. E.g., given two utterances *"... auf dem Stuhl liegt ein Paket, und vor dem Stuhl steht ein Paket"* and *"... und einmal von dem grünen Fass, was hier liegt"* both with the target *'liegt'*, it is unlikely that knowing that *'liegt'* is a verb and *'Paket'* and *'Fass'* are nouns can help us identify that the correct frame is *'Presence'* in the first case and *'Posture'* in the second one. As for the simple errors made by the classifier with POS tags, their number stayed the same: the classifier was actually able to assign the right frames to 6 out of 14 utterances misclassified by the basic model, but at the same time it made 6 new errors. Lastly, two errors caused by the wrong annotation/preprocessing, have not been corrected either. So, based on the mistake analysis we can conclude that the influence of the POS feature is not really big, and the IBA increase by 3% - 4% is very likely to be determined by the stochasticity of the training procedure. Especially, if we consider that the model was tested on a rather small test set.

As Table 4.10 shows, extending token embeddings with subword mask embeddings as well as using the combination of two extra features likewise does not seem to really influence the performance of the classifier. Note that in case of the frame classifier trained on the English TRADR dialogues, the effect of additional features was also rather insignificant. In relation to this we should mention that recently there appeared several studies that investigate how attention in BERT works and what language patterns it is able to learn. These studies actually shed some light on why adding lexical, syntactic or other features to token embeddings has little to no effect. E.g., Clark et al. [2019] investigated the behaviour of all 144 BERT's attention heads and their ability to classify various syntactic relations. They found out that certain heads can identify direct objects of verbs, determiners of nouns, objects of prepositions, and objects of possessive pronouns with surprisingly high accuracy. Htut et al. [2019] report similar observations. So, even if none of BERT's attention heads incorporates all syntactic

information learned by the model, it looks like BERT is still able to learn quite a lot about various dependency relations in the data, so that the information about POS tags and subword masks gets to certain extent redundant and does not produce the expected effect.

Finally, we would like to mention one more modification of the basic model. It assumes extending token embeddings with POS tag embeddings, because they still seem to be more useful than subword mask embeddings, and using the equal sampling from the SALSA dataset, which showed better results than the balancing one. The IBA score is given in Table 4.10, and it is rather perplexing. We see that the equal sampling, which earlier helped us achieve the IBA score of 90%, has not provided the anticipated positive effect - the IBA is 84% for both test sets, so that it is only slightly better than the score of the basic model.

Trying to understand the difference in scores, we again refer to Table 4.11 which summarizes the performance of different folds as well as the averaged validation IBA scores of the two models with POS tags during the 5-fold cross-validation procedure. In this table, namely in columns four and five, we see that the averaged IBA results for both models are the same, and they are by 1% smaller than the averaged scores of the models without the POS tag feature. The scores actually support our hypothesis that adding lexical features to token embeddings does not have any real positive influence on the model. Moreover, adding such features seems to confuse the frame classifier, so that sampling looses its positive effect on the accuracy.

On the other hand, while the averaged validation IBA score of the basic model with POS tag feature and the model with POS tag feature and equal sampling is 83%, in Table 4.10 we see that both models achieve slightly better results on the test sets - 85% and 84%, respectively. This can probably be explained by the fact that according to our approach, the models with the best validation scores across the five folds were chosen, and as a result the test IBA turned out to be better than the averaged validation one.

In conclusion we would like to summarize our observations during the training and testing of different versions of frame classifier for both English and German data. First, the PAFIBERT model with its position-based attention mechanism suggested by Tan and Na [2019] was able to achieve rather good IBA when trained on the TRADR dialogue data, despite the small size of the training sets. Second, we saw that sampling additional training instances from unrelated domains can be useful only if the original training set is small enough, but as it grows (even insignificantly) the positive effect of sampling decreases. Also, sampling seems to be beneficial for handling simple errors, but rather ineffective for cases that require disambiguation. Third, we found no significant effect of adding lexical and/or discourse features to token embeddings. In addition, a combination of sampling with an extra POS tag feature even seems to neutralize a positive effect of sampling. However, here we must note that such observations need to be additionally checked by training and testing a model on much larger datasets. Fourth, we came to the conclusion that applying filtering can also be beneficial. It helps exclude simple errors, but also requires much more training data, because such an approach can only be effective if the model has a good coverage. Finally, the analysis of mistakes made by various versions of the frame classifier for both English and German showed that most of them happen because of ambiguous targets.

They are especially difficult to learn, and adding more training data does not seem to help. Speaking of targets, it is also necessary to mention that multi-token target expressions incorporating context turned out to be less effective than single-token ones, as the former may confuse the classifier.

Chapter 5

Conclusion and Future Work

In this chapter we will recap our most significant achievements and findings, and present some ideas concerning possible future work.

In this thesis we investigated the possibility of using frame semantics as a means of providing a meaning representation framework for English and German dialogue data coming from the domain of robot-assisted disaster response. We found semantic frames to be easy and convenient for capturing the meaning of an utterance depending on the target expression - the approach is span-based and does not require any sophisticated data annotation or preprocessing. We successfully reused the PAFIBERT frame classifier developed by Tan and Na [2019] for our data and were able to achieve the IBA score of 88% - 90% on the test sets. Despite the fact that we had much less data than Tan and Na [2019], who trained their models on the much larger FrameNet corpus, our scores are comparable with theirs. As far as we know, our work is the first attempt to use semantic frames for the domain of disaster response, and is among a few approaches which aim at implementing a frame classifier for dialogue data.

In addition, we investigated the impact of sampling additional training instances from an unrelated domain on the classifier's performance. We found that sampling helps achieve better IBA scores if the original training set is small, but as the size of the training set increases, the positive effect of sampling quickly disappears. So, in case of the German data with about 16,600 training tokens, the effect of sampling was noticeable, however for the English data with more than 21,200 training tokens sampling did not lead to any improvement. To strengthen our conclusion about the negative correlation between the size of the original training data and the effect of sampling, some additional experiments could be useful. E.g., it would be interesting to check whether sampling will have any positive effect, if we decrease the size of the English training data. In this thesis we tried out various types of sampling, namely sampling certain percentage of data, balancing sampling, which aims at making the training data more balanced, and equal sampling, which does not allow the number of additional instances be larger than the number of the original ones in the same class. It turned out that equal sampling was slightly more effective than two other types. We did not perform any experiments with over- and/or undersampling which implies sampling from the original dataset and is often used with imbalanced data. This can be a subject for further research, especially interesting is an approach that assumes generating synthetic training instances, e.g., the embeddings incorporating the targets

and their context.

Besides sampling, we also examined how extending BERT token embeddings with lexical and discourse features influence the model's performance. The lexical features included POS tags and subword masks of the tokens, and discourse features - the speaker and dialogue act tags. In contrast to our expectations, both lexical and discourse features failed to demonstrate any stable positive influence on the model's accuracy. Moreover, the validation and test IBA scores of some models with certain features, as well as the scores of English and German models with the same extra features were contradictory, e.g., the German model with POS tags had a worse validation IBA than the basic model, but a better test one, and the combination of POS tags and subword masks seemed to work for the English model, but was the least successful for the German one. Judging by such results we have to conclude that more training and test data, as well as significance tests are required to estimate the impact of extra features correctly.

For the English frame classifier we additionally performed experiments with s varying context window size and a frame filtering mechanism. The experiments revealed that varying the context window size does not have much influence on the IBA score, and filtering can be beneficial in case a much larger training set with a better coverage of targets (lexical units) is available.

In general we saw that the described modifications had slightly different impact on English and German frame classifiers. We are inclined to think that a larger test set is needed to draw clearer conclusions. It should also be noted that in order to train all presented frame classifiers we used the 5-fold cross-validation procedure. Out of five models we only saved the one that showed the best results on the validation set and then evaluated this model on the test sets. However, in the future it could also be interesting to check whether it is possible to achieve better performance using the ensembling technique, i.e. save all the five models and combine them into a single one.

Also, we studied the mistakes made by different versions of both English and German frame classifiers. As expected, the largest group of mistakes is represented by ambiguous targets, many of which evoke semantically close frames. The problem is often aggravated by the fact that either the context does not contain any clues that would allow to disambiguate such targets, or the necessary clues are part of previous utterances and are difficult to retrieve. The problem of disambiguation undoubtedly requires more research if we want to improve the performance of the model. Besides this, we found that, in comparison to the basic English frame classifier, the German one makes more simple errors that have nothing to do with target ambiguity. Obviously, this happens due to a smaller size of the German training data, as sampling helps to considerably decrease the number of such errors.

For now we focused only on automatic frame assignment given a target, and did not pay any attention to the elliptical utterances where the targets are omitted. However, as the analysis of the TRADR data showed, elliptical utterances make up almost 19% of the English TRADR corpus and more than 10% of the German one. As ellipsis is very typical of dialogue, it is necessary to research possible approaches to inferring the omitted parts of elliptical utterances.

Also, in the given thesis we did not perform other steps that are typically associated with the frame semantic parsing. These steps include automatic target identification,

and recognition and classification of arguments (frame elements), and can be subjects of future work. E.g., for target identification it is possible to use hand-crafted rules picking targets depending on POS tags or tokens' positions within a dependency tree. Another opportunity is to train a classifier to choose the most semantically important tokens, or simply to decide whether a given token can be a target. Identification and classification of frame elements are considered to be more challenging tasks, and one of the latest approaches in this area implies merging them into a single sequence-to-sequence generation problem. This approach was suggested by Kalyanpur et al. [2020] who performed their experiments on the text data, however, it can be easily transferred to any domain including ours.

In addition, we took into consideration the fact that our training and test datasets were imbalanced and used the appropriate metrics to measure the performance of the classifiers. While many researchers working with the FrameNet data used standard accuracy to evaluate their models (e.g., the absolute majority of those mentioned in Section 2), we relied on the index of balanced accuracy metric (IBA) which tries to consider the contribution of all classes to overall performance and is able to regulate the influence of the majority classes. In addition to the IBA metric our classification reports also include sensitivity, specificity and geometric mean which are also suitable for imbalanced data, as well as precision and F-score which are less reliable for imbalanced data, but nevertheless are widely used.

Another important achievement is the annotation of both English and German TRADR data with semantic frames and frame-evoking targets (frame related elements). In total, the English TRADR corpus contains 3,521 frame instances, and the German one - 3,519. We summarized our annotation approach as annotation guidelines which can be found in Appendix A and defined ten new frames presented in Appendix B. As far as we know, the TRADR corpus is the only corpus annotated with frames in the domain of team communication in disaster response. For the tasks of argument recognition and classification the corpus can be further annotated with frame elements.

Not only did we annotate the TRADR corpus with targets and frames, but we also examined it in detail. We described the distributions of dialogue turns and semantic frames, as well as that of lexical units that correspond to the targets and their parts of speech. We analysed the utterances from the point of view of completeness and ambiguity, and discussed the main topics and characteristics of team communication. We also performed a similar analysis of FrameNet data and compared the two datasets. Our analysis can be helpful for other researchers who plan to use FrameNet data for training or sampling, or want to adapt the models trained on FrameNet data for other tasks.

Appendix A

Annotation Guidelines

In what follows, we will present the annotation guidelines that were worked out during the annotation of English TRADR data with semantic frames. All examples are taken from the TRADR data, except another source is given. Words in bold stand for frame-evoking targets.

A.1 Preliminaries

For now, the annotations are stored in a simple *Excel* file. Each dialogue takes a separate *Excel* sheet. The data was previously annotated with dialogue acts including dimensions and corresponding communicative functions according to the ISO standard 24617-2 [Bunt, 2019]. These annotations are complemented with information about semantic frames. We add the following columns to the existing annotations: 'Frame ID', 'Frame', 'Parent frame', 'LU' and 'Target'. Each annotation instance takes its own row. Frame labels are taken from the FrameNet 1.7 database. Currently, we do not annotate frame elements, grammatical functions and phrase types. Where possible, we try to rely on the FrameNet annotation guidelines [Ruppenhofer et al., 2006].

A.2 Annotation unit

We regard a single utterance, including incomplete ones, as a separate annotation unit. Because the data was earlier annotated with dialogue acts, several utterances may be treated as a single dialogue act, if they fulfill the same function. In such cases we need to split them (see Example A.2.1).

Example A.2.1. *Dialogue act splitting*

Dialogue act annotation:
[TASK/INFORM TL: Over here. There we have two victims. One victim is a saved.]

Frame annotation:
(a) [LOCATIVE_RELATION Over here].
(b) [PRESENCE There we have two victims].
(c) [RESCUING One victim is a saved].

On the other hand, some utterances in the original data were split, either because they contained two or more different dialogue acts, or because dialogue acts were the same, but the described actions were different. Such utterances are merged back to their original forms (see Examples A.2.2 and A.2.3).

Example A.2.2. *Dialogue act merging (different dialogue acts)*

Dialogue act annotation:
(a) [TASK/INFORM UGV-1: I put an icon on the plant where I saw fire.]
(b) [COMMUNICATION MANAGEMENT/SELF-CORRECTION UGV-1: I was... saw]

Frame annotation:
[CREATE_REPRESENTATION I put an icon on the plant where I was... saw fire.]

Example A.2.3. *Dialogue act merging (same dialogue acts)*

Dialogue act annotation:
(a) [TASK/REQUEST TL: Can you go toward the red barrel]
(b) [TASK/REQUEST and see what in it is?]

Frame annotation:
[CAPABILITY Can you go toward the red barrel and see what in it is?]

A.3 Nested frames

We assume that a single utterance may contain more than one frame. These frames may be nested (see Example A.3.1. Our annotation format suggests using a separate row for each frame, therefore, in this case we create several copies of the utterance - a copy for each frame.

The reason for such an approach is that often it is impossible to pick out a single frame that would be semantically the most important. However, if we assume potential existence of multiple targets in a sentence, then we need some criteria for choosing the semantically relevant ones, because theoretically we can treat each meaningful word as a frame-evoking element.

Example A.3.1. *Nested frames*
[CAPABILITY And I can not [PERCEPTION_EXPERIENCE see [BREATHING he is breathing.]]]

A.4 Target annotation

In this section we will present the main criteria defining the choice of frame-evoking targets.

We treat verbs as main frame-evoking elements. In cases, when the verb form includes auxiliary verbs, we mark only semantically important element, i.e. the main verb, as target (see Example A.4.1).

Example A.4.1. *Main verb as target*
*[SENDING Pictures have been **sent**.]*

Usually a target is a single token, even when a word form consists of several ones. There are, however, some exceptions. First, we treat phrasal verbs or multi-word idiomatic/fixed expressions as a whole as targets (see Examples A.4.2 and A.4.3). Second, we consider expressions consisting of a support verb and a noun, which is a semantic head in this case, to be multi-word targets (see Example A.4.4).

In FrameNet terminology, a support element (it is not necessarily a verb) is an element that syntactically governs the semantic head, but does not reliably represents the same meaning if taken separately [Ruppenhofer et al., 2006].

It is important to note that in this very case the FrameNet annotation guidelines treat only semantic heads as targets. Our approach is different, because we give more weight to verbs, as we plan to correlate frame annotations with semantic role labelling in the future.

Example A.4.2. *Phrasal verb as target*
*[BECOMING_AWARE Yes, I try to **find** that **out**.]*

Example A.4.3. *Fixed expression as target*
*[HOLDING_OFF_ON I **put** my robot **on hold**.]*

Example A.4.4. *Support verb plus noun as target*
*[COMMUNICATION_RESPONSE I can't **give the answer**.]*

In many cases elements that reflect the meaning of an utterance best are not verbs. They can be nouns, adjectives, prepositions and so on. This happens, for instance, if we deal with linking verbs or expletives. So, in Example A.4.5 both targets are adjectives, and in Example A.4.6 one target is an adjective and the other is a noun.

Example A.4.5. *Linking verbs*
*(a) [AWARENESS Number of missing people is **unknown** too.]*
*(b) [CORRECTNESS That's **correct**.]*

Example A.4.6. *Expletive constructions*
*(a) [POSSIBILITY Is it **possible** to take a picture from above and to send it to me, UAV drone?]*
*(b) [EXPLOSION There was an **explosion** in the area between the two furnaces.]*

Sometimes, even if a verb functions as a normal meaningful element, it does not necessarily represent the semantics of an utterance. As shown in Example A.4.7, the location of the robot is more important than the fact that it is 'standing' and not, say, 'lying'. And the noun 'possibility' reflects the meaning of the question better than the verb 'have'.

Example A.4.7. *Non-verbal targets*
*(a) [LOCATIVE_RELATION I am standing **in front of** one.]*
*(b) [POSSIBILITY Do you have the **possibility** to change the level?]*

We also treat participles I (gerunds) and II (past participles) as potential targets, because they are verbal forms and often play an important role in the meaning of an utterance, as Example A.4.8 demonstrates.

Example A.4.8. *Participles as targets*
*(a) [SCRUTINY Go on **searching**.]*
*(b) [FLUIDIC_MOTION When I turned around the vessel there was a light liquid substance **leaking**.]*
*(c) [CREATE_REPRESENTATION I'd need some pictures from there, **taken** from the middle of the two furnaces and around the furnace.]*

A.5 Lexical unit annotation

Whereas a target is a direct constituent of a sentence, i.e. a word form, a lexical unit is a lemma of this word form. This means that each lemma can have several different instantiations. Each lemma is complemented with a tag representing its part of speech (see Example A.5.1). For the full list of possible tags, please, refer to the FrameNet annotation guidelines [Ruppenhofer et al., 2006] or the corresponding website [FrameNet, 2020]. We store lexical units in a separate column.

Example A.5.1. *Targets and LUs*

Target	*LU*
leaks	*leak.v*
leaked	*leak.v*
taking a picture	*take a picture.v*
correct	*correct.a*
roger	*roger.intj*

For now we mostly use information about lexical units in order to learn more about the distribution of our data, namely what lemmas are the most/least frequent, percentage of polysemous lemmas, etc.

A.6 Frame annotation

Here we will introduce our main frame annotation principles.

A.6.1 General vs. specific frames

It was already mentioned that our goal is to find a frame that is semantically the closest to the given utterance. However, as we do not have much data, such an approach may easily lead to a situation, when we have only a couple of instances of certain frames, and this is not enough to train a reliable frame classifier. This may happen if a frame has many subframes. For example, a frame *'Locative_relation'* has 14 child frames, such as *'Abounding_with'*, *'Adjacency'*, *'Directional_locative_relation'*, *'Expected_location_of_person'* and so on. A frame *'Motion'* has 5 child frames, some of which have further subframes. In such circumstances a rule of thumb is to choose a frame that is neither too general nor too specific (see Example A.6.1).

Example A.6.1. *Preference to more general frames*
*(a) [LOCATIVE_RELATION It was near the furnace on the left **at** a height of about 50*

meters.] (not 'Spacial_ co-location' frame)
*(b) [LOCATIVE_RELATION Should be right **next to** the barrel.] (not 'Adjacency' frame)*
*(c) [LOCATIVE_RELATION That's **inside** the plant.] (not 'Interior_profile_relation' frame)*
*(d) [MOTION Can you describe exactly in which direction you want me to **fly**?] (not 'Self-motion' frame)*

On the other hand, sometimes the nuances of meaning of parent and child frame are too important to merge the two frames. In such cases we keep them separate. As shown in Example A.6.2, the target *'guide'* evokes the frame *'Cotheme'*, not *'Self-motion'*, which is the parent frame of *'Cotheme'* and not *'Motion'*, which is the parent frame of *'Self-motion'*. And in the second sentence we also think it is necessary to capture the differences between motion in general and motion of a fluid.

Example A.6.2. *Preference to more specific frames*
*(a) [COTHEME Could you **guide** UGV 2 to that place?] (not 'Self-motion' or 'Motion' frames)*
*(b) [FLUIDIC_MOTION Nothing **leaking** from the barrel.] (not 'Motion' frame)*

A.6.2 Adapting existing frames

Next frame annotation principle has to do with the fact that the domain we work with is rather specific. As a result, some domain-specific frames are missing in the FrameNet database. Sometimes, however, it is possible to reuse certain frames so that they cover new phenomena.

Below we will present some guidelines taken from Ruppenhofer et al. [2006], which tell us when it is possible to use an already defined frame:

- The meaning of the sentence in question is semantically close to the meaning captured by a candidate frame.
- The number and types of explicit and implicit frame elements of a candidate frame and in the given sentence are equal.
- There should be aspectual coherence, i.e. if a candidate frame depicts an event that consists of several sub-events/stages, the lexical unit in question must describe an event with the same structure.
- The perspectives (e.g. *'Buyer'* vs. *'Seller'*) should be the same.
- Semantic relations between frame elements and to other frames should be the same for all lexical units in the frame.

If these criteria are not met, and some systematic differences can be detected, the existing frame should be split in two or more frames, or a new one should be introduced.

Here are some other criteria, also from Ruppenhofer et al. [2006], that allow us to reuse already existing frames:

- Grammar differences, such as voice, aspect, tense, etc.
- Antonymy (e.g. *'high'* and *'low'* both evoke *'Position_on_a_scale'* frame)

- Usage differences due to deixis (*'come'* vs. *'go'*), register (slang vs. formal language), dialect, evaluation (*'criticize'* vs. *'praise'*), etc.

Example A.6.3 demonstrates the principle of reusing/adaptation. So, we take the *'Create_representation'* frame to describe the action of creating of a point of interest on the screen. Similarly, we use *'Being_located'* frame not only to speak about something that is in stable position with respect to some location, as the original definition says, but also to describe positions that can change, but are stable at the current moment, for instance, a position of a robot or a victim. Some jargon words/expressions, like *'code three'* (i.e. *'dead'*) can also be mapped to existing frames.

Example A.6.3. *Reusing frames to cover new phenomena*
*(a) [*CREATE_REPRESENTATION *Can you **make a POI** of that bottle?*
*(b) [*BEING_LOCATED *Does the picture show me where it is **located**?]*
*(c) [*DEAD_OR_ALIVE *The victim is **code three**.]*

Sometimes, when we have difficulties finding a suitable frame in the database by the given lexical unit, paraphrasing the utterance seems to be a solution of the problem. However, according to the FrameNet annotation guidelines [Ruppenhofer et al., 2006], in most cases it is not so. Generally, only paraphrasing of separate lexical units works, but not paraphrasing of the whole sentences. The latter changes sentence structure, and as a result the distribution of meaning across the sentence parts changes too (see Example A.6.4 from Ruppenhofer et al. [2006]).

Example A.6.4. *Paraphrasing changes meaning*
*(a) [*CHANGE_OF_CONSISTENCY *The paste **hardened** due to hydration of the cement.*
*(b) [*CAUSE_CHANGE_OF_CONSISTENCY *The hydration of the cement **hardened** the paste.]*

Example A.6.5 shows two utterances, where we tried to paraphrase the lexical units. In both cases the target verbs *'give'* and *'relate'* are present in the database, but not in the senses we need.

Example A.6.5. *Paraphrasing of lexical units*
*(a) [*COMMUNICATION *Could you **give** me further details on the container?] ('give' ≈ 'communicate')*
*(b) [*GRASP *To be able to **relate to** the location of the barrel.] ('relate' ≈ 'understand')*

A.6.3 Introducing new frames

Still, in some cases it is impossible to adapt to our purposes frames already existing in the FrameNet database. Moreover, sometimes certain word senses are not present in the database at all, as it is not exhaustive. In such situations we need to introduce new frames.

First of all, we work with a very specific domain: team communication in disaster response. This means, there exists a certain communication protocol that the mission participants should follow. There are certain phrases to be used in order to establish

communication, check the connection quality, accept or reject a mission request and so on (see Examples A.6.6 and A.6.7). The FrameNet database does not have such specific frames.

Example A.6.6. *Communication by protocol*
*(a) [*COMMUNICATION_PROTOCOL *Team leader for operator one.]*
*(b) [*COMMUNICATION_PROTOCOL *Team leader* ***listening****.]*
*(c) [*COMMUNICATION_PROTOCOL ***Give message****.]*

Example A.6.7. *Communication response message*
*(a) [*COMMUNICATION_RESPONSE_MESSAGE *Yes,* ***roger****.]*
*(b) [*COMMUNICATION_RESPONSE_MESSAGE ***Negative****.]*
*(c) [*COMMUNICATION_RESPONSE_MESSAGE *Yes,* ***positive****.]*

Note that, in principle, we can treat such utterances as 'normal' ones and assign the corresponding frames from the FrameNet database. So, the utterance *"Team leader* ***listening****"* would get the label *'Perception_active'*, and the utterance *"****Negative****"* - the label *'Attitude_description'*. However, we decided against such an approach, and introduced new frames in order to emphasize the specific nature and purpose of such utterances.

Instances of *'Communication_by_protocol'* frame can be easily recognized, as they normally contain mentions of mission roles (e.g., *'team leader'*, *'operator one'* or 'UAV') and targets that relate to communication (e.g., forms of *'listen.v'*, *'answer.v'*, *'come in.v'*, etc).

We also use the frame *'Communication_response_message'* for all short answers, positive, as well as negative.

Annotating dialogues we often come across incomplete utterances. If an incomplete utterance has some meaningful elements, so that we are able to assign frames to them, we do so (see Section A.6.4 for details). However, often such utterances only contain different noises, false starts, repetitions and make little sense. We use a new frame called *'Communication_fragment'* for such cases (see Example A.6.8).

Example A.6.8. *Communication fragment*
*(a) [*COMMUNICATION_FRAGMENT *Erm...]*
*(b) [*COMMUNICATION_FRAGMENT *We- you]*
*(c) [*COMMUNICATION_FRAGMENT *Well, it is...]*

Missing frames are not always domain-specific. So, Example A.6.9 shows three new frames that we had to introduce, because we could not find suitable frames in the FrameNet database. When a new frame is being introduced, it is important to check, if we can integrate the frame in question into the existing frame hierarchy, i.e. find potential parent and sibling frames. For instance, our new frame *'Level_of_clarity'* has parent frame *'Gradable_attributes'*, and frame *'Be_piece_of'* is a child frame of *'Being_included'*.

Example A.6.9. *Word senses absent in FrameNet database*
*(a) [*LEAD *The stairwell* ***leads*** *upwards.]*
*(b) [*LEVEL_OF_CLARITY *Yes, the pictures aren't very* ***sharp****.]*

*(c) [BE_PIECE_OF I can also see fragments that **belong to** the building lying around here.]*

Finally, we would like to give one more example showing that paraphrasing does not always work - Example A.6.10. In both cases demonstrated here we had to come up with new frames, namely *'Level_of_substance'* and *'Being_reasonable'*, because the paraphrases and the corresponding frames that are available in the FrameNet database represent meanings different from the original ones. So, *'smoke'* denotes *'substance'*, while *'smoky'* also incorporates the degree. And while *'reasonable'* in the sense present in the FrameNet database describes a person and their behaviour, *'make sense'* cannot be applied to a person, in the current context the expression means that a certain decision is rational and practical.

Example A.6.10. *Paraphrasing fails*
Original utterance:
*[LEVEL_OF_SUBSTANCE It's actually quite **smoky**.]*

Paraphrase:
*[SUBSTANCE There's actually lots of **smoke**.]*

Original utterance:
*[BEING_REASONABLE It would definitely **make sense** if you let the UAV guide you.]*

Paraphrase:
*[MENTAL_PROPERTY It would definitely be **reasonable** if you let the UAV guide you.]*

The full list of new frames that we introduced while annotating TRADR dialogues is given in Appendix B.

A.6.4 Ellipsis

One more important annotation principle has to do with annotation of elliptical sentences. In contrast to communication fragments described in Section A.6.3, elliptical utterances are absolutely meaningful, with some parts omitted, because the dialogue participants can easily infer them from context.

In cases of ellipsis we always try to mentally 'restore' an incomplete utterance using its immediate context (i.e. two or three previous utterances) and infer the right frame (see Example A.6.11). We place a label 'E' in both 'LU' and 'Target' columns to mark such utterances.

Example A.6.11. *Frame inference in elliptical utterances (targets omitted)*
Original utterance:
[IDENTITY Kind of a sidewalk.]

Inferred utterance:
*[IDENTITY It **is** a kind of a sidewalk.]*

Original utterance:
[PERCEPTION_EXPERIENCE Like a third barrel that could be around.]

Inferred utterance:
*[PERCEPTION_EXPERIENCE Can you **see** anything like a third barrel that could be around?]*

It is important to mention that sometimes an utterance can be elliptical from the point of view of linguistics, however we do not label it as such. This happens if the target is not omitted, and the corresponding frame can be assigned (see Example A.6.12).

Example A.6.12. *Elliptical utterances (targets present)*
Original utterance:
*[MOTION Is **going** away.]*

Inferred utterance:
*[MOTION The victim is **going** away.]*

Original utterance:
*[CREATE_REPRESENTATION I have **made a picture** with red.]*

Inferred utterance:
*[CREATE_REPRESENTATION I have **made a picture** with red barrel.]*

We do not treat short positive or negative answers like *"Yes"* and *"No"* as elliptical, even if according to some classifications they are labeled as such. We do so, because in many cases short answers relate to several previous frames, so we would have to infer all of them at the same time. Therefore, short answers are normally labelled with *'Communication_response_message'* frame (see Example A.6.13).

Example A.6.13. *No inference for short answers*
*TL: [MOTION **Fly** westwards], [ATTEMPT_SUASION **try** to find the victim],*
*[CREATE_REPRESENTATION if you see something interesting, **make a picture**.]*
*UAV: [COMMUNICATION_RESPONSE_MESSAGE **Yes**.]*

A.7 Parent frames

For each frame assigned to an utterance we try to find the corresponding parent frame in the FrameNet database using the relation *'Inherits from'*. A frame can inherit from more than one frame. In this case we place all given parents into the corresponding slot. We are interested only in immediate parent frames, i.e. we do not search further up to the root frame. If the given frame is an independent one, i.e. has no parent(s), we store its label as a parent. If we introduce a new frame, we try to find a place for it in the FrameNet hierarchy (see Section A.6.3.) Example A.7.1 shows some parent frames.

Example A.7.1. *Frames and Parent frames*

Frame	*Parent frame*
Sending	*Sending*
Motion	*Event*
Communication	*Cause_to_perceive*
Capability	*Gradable_attributes, Possibility*
Locative_relation	*Basis_for_attribute, State*

Parent frames can be useful in case one needs to group certain frames together in order to decrease the number of labels while training a classifier.

A.8 Annotation issues

In what follows we will present main challenges that we faced with while annotating the TRADR data, as well as some suggestion about how to deal with them.

A.8.1 Absence of lexical units and senses in the database

One of the most obvious problems is absence of certain lexical units and word senses in the FrameNet database. If a lexical unit is missing, we check if its synonyms are present, or if there exist a close frame that could be adapted (see Section A.6.2 for details). A related issue here is that sometimes it is very difficult to find the necessary frame in the database even if it actually exists. If neither the lexical unit nor the word sense can be found, a new frame is introduced (see Section A.6.3).

A.8.2 Disagreement between the meaning of an utterance and the frame evoked by the target

However, sometimes both the lexical unit and the corresponding frame(s) can be found, and still there is a feeling that they do not really represent the meaning of the utterance. Often, especially when the targets are some simple, common words, like *'have'*, *'give'* or *'take'*, the frames found in the database seem to be either too general, or just not really suitable. The most typical usages of these words in our domain relate to communication between mission participants, distribution of tasks, information exchange, etc.

Example A.8.1 is a good illustration of the problem. According to the FrameNet database, the lexical unit *'give.v'* evokes three frames: *'Giving'*, *'Infecting'* and *'Sex'*. Out of them only the frame *'Giving'* looks like a possible candidate to label the sentences *(a) - (c)*. The problem is that *'Giving'* corresponds to the literal meaning of *'give.v'*, i.e. describes an event, when a donor transfers some object to a recipient. And the real meanings of the utterances *(a) - (c)* have very little to do with this sense of our target. Similarly, the *'Possession'* frame, evoked by the lexical units *'have.v'* and *'have got.v'*, is not the perfect choice for sentences *(d) - (e)*, because it represents the basic, literal meaning of both verbs.

In other words, the dilemma is as follows. We can either reuse the FrameNet frames interpreting their definitions very loosely and metaphorically, or we can find more suitable frames, which would have very little to do with our targets.

Example A.8.1. *Domain-specific word senses*
*(a) [*IMPOSE_OBLIGATION *I* ***gave*** *you this area.]*
*(b) [*COMMUNICATION *Can you* ***give*** *me your status?]*
*(c) [*COMMUNICATION_BY_PROTOCOL *Team leader,* ***give message****.]*
*(d) [*PRESENCE *I don't* ***have*** *any percentage.]*
*(e) [*POSSESSION *I* ***'ve got*** *an arm with me.]*

As Example A.8.1 shows, we prefer finding more suitable frames to metaphorical interpretations. Only in *(d)* we had to reuse *'Possession'* frame, as we could not find a better candidate in the FrameNet database.

A.8.3 Distribution of meaning within the utterance

There is another issue related to the one described above. Namely, if our domain and immediate context have such big influence on the interpretation of our targets, does not this mean that they actually perform a supportive function? Would it make sense to use the whole phrase as a target, e.g., *'give status.v'*, or pick up other targets if possible?

This question is not easy to answer. Have a look at Example A.8.2. All utterances here have something to do with communication/information exchange, and all of them have *'give'* as as main predicate. While it looks reasonable to make such words as *'feedback'*, *'details'* or *'answer'* be part of the target (see ceases *(c)* - *(e)*), including *'procedure'* or *'location'* into targets does not really make much sense, as these words have nothing to do with communication (see cases *(a)* - *(b)*).

Example A.8.2. *Multi-word targets?*
*(a) [*COMMUNICATION *It's procedure I* ***gave*** *you.]*
*(b) [*COMMUNICATION *Can you* ***give*** *me your location?]*
*(c) [*REPORTING ***Give*** *me some* ***feedback****.]*
*(d) [*COMMUNICATION *Can you* ***give*** *me some* ***details*** *on where the smoke development is?]*
*(e) [*COMMUNICATION_RESPONSE *I can't* ***give*** *the* ***answer****...]*

Another option would be to treat objects of *'give'* as targets. But while this approach would work for utterances *(c)* and *(e)* of Example A.8.2, in cases *(a), (b)* and *(d)* that would be clearly wrong. In general, our solution is to extend the targets, if it helps to represent the meaning of an utterance better.

Example A.8.3 illustrates another aspect of the problem discussed, namely the difficulty to keep the annotations consistent. We use *'Create_representation'* frame to label two activities typical for our data: taking a photograph and marking an object on the computer screen with a special symbol. So, according to our annotation approach, we extend our targets, as utterances *(a)* and *(b)* demonstrate. But the situation with creating *'a point of interest'* on the screen is more difficult. Similarly to *'make a picture.v'*, we include *'a point of interest'* to be part of the target, as in utterance *(c)*. However, we can notice that with other verbs rather than *'make'*, like *'place'*, *'put'* or *'mark'*, the part *'a point of interest'* is often omitted, as in sentence *(d)*, so that we are left with the verb only.

Example A.8.3. *Multi-word targets?*
*(a) [*CREATE_REPRESENTATION *Please,* ***give*** *me a total* ***overview*** *of the central building.]*
*(b) [*CREATE_REPRESENTATION *I* ***take a snapshot*** *but I don't know how to share it.]*
*(c) [*CREATE_REPRESENTATION ***Make a point of interest*** *of it.]*
*(d) [*CREATE_REPRESENTATION *Yes, I've* ***placed*** *an obstacle I can't go through because I'm too high.]*
*(e) [*CREATE_REPRESENTATION *I* ***placed*** *victim point of interest on the map.]*

The current solution to the problem is to use a convention, according to which we normally extend certain verbal targets like *'make'* or *'take'*, and do not extend others, such as *'place'* (compare utterances *(c)* and *(e)* in Example A.8.3).

This situation is also complicated by the fact that, according to the definition of *'Create_representation'* frame given in [FrameNet, 2020], such elements as *'a snapshot'*, *'picture'*, *'point of interest'*, etc. can be interpreted as a core frame element called *'Representation'*.

A.8.4 Target choice

There is one more issue related to the distribution of meaning within an utterance, namely, the choice of target. The problem discussed in Section A.8.3 was as follows: we know the main meaning of an utterance, but we are not sure what elements represent it best. Here the problem is that it is difficult to decide what aspect of utterance meaning is the most prominent.

Example A.8.4 illustrates the situation. All utterances *(a)* - *(e)* have similar structure with *'have'* as the main verb. However, the frames assigned to these utterances are all different. In principle, it is possible just to follow our strategy that says to focus more on verbs (see Section A.4), and assign the frame *'Presence'* to all of the utterances. But does the frame *'Presence'* represent the meaning of the utterances well? Let us see. In sentences *(a), (d)* and *(e)* *'task'*, *'problem'* and *'possibility'* seem to be more prominent than *'have'*. Moreover, they stand for abstract notions that cannot really be present at any location.

On the other hand, choosing the word *'picture'* instead of *'have'* as target in *(c)* would evoke the frame *'Physical_artwork'*, which does not say anything about who has got the picture and for whom. So, in this case the frame *'Possession'* seems to be more relevant. The situation is similar with sentence *(b)*.

Example A.8.4. *What is the meaning?*
*(a) [IMPOSING_OBLIGATION I have a **task** for you.]*
*(b) [PRESENCE They didn't **have** enough air.]*
*(c) [POSSESSION I **have** a nice picture for you.]*
*(d) [PREDICAMENT I have a **problem** with my robot.]*
*(e) [POSSIBILITY Do you have the **possibility** to change the level?]*

In general, it is rather difficult to make up a set of simple rigid rules saying how to choose a target so that it reflects the main meaning of the utterance. The choice can be influenced by context, and is often subjective, as different people may interpret the same sentence differently.

A.8.5 Frame choice

Sometimes we have several similar candidate frames for an utterance. Often, there is only slight difference between such frames, which is difficult to detect in certain contexts. This often happens when the frames are tied to each other via *'see Also'* relation. The FrameNet annotation guidelines state that *'see Also'* relation has no

semantic meaning, it connects two confusable frames which have overlapping lexemes [Ruppenhofer et al., 2006].

Example A.8.5 illustrates the issue. According to their definitions (see [FrameNet, 2020]), both *'Inspecting'* and *'Scrutiny'* frames describe an event when an *'Inspector'/'Cognizer'* is paying close attention to some *'Ground'*. The difference is that *'Inspecting'* assumes checking if the *'Ground'* is intact or if some *'Unwanted entity'* is present, and *'Scrutiny'* focuses on discovering of the salient characteristics of the *'Ground'*, or if some *'Phenomenon'* is present/contained in the *'Ground'*. However, in our context the salient characteristics and phenomena often turn out to be unwanted entities, or the existence of such entities is implied, but never mentioned explicitly.

So, we label utterances *(a)* and *(b)* with the frame *'Inspecting'*, because the *'Unwanted entities'* are given explicitly, and assign the frame *'Scrutiny'* to utterances *(c)* and *(d)*, because potential entities are only implied in *(c)* and in *(d)* the *'victim'* is interpreted as a *'Phenomenon'* and not as an *'Unwanted entity'*. However, note that the difference in meaning between all these sentences is really superficial, and the assignment of frames is only the matter of interpretation.

Example A.8.5. *Inspecting or Scrutiny?*
*(a) [INSPECTING Please, also **check** with the infrared camera if there are any people down there.]*
*(b) [INSPECTING Could you move southwards and **check** the building for dangerous substances?]*
*(c) [SCRUTINY So, you've basically **checked** the actual area around the furnaces?]*
*(d) [SCRUTINY Please go into the plant and **search** for victim...]*

Frames *'Existence'* and *'Present'* also represent a source of confusion. The former simply declares an existence of something irrespective of time, duration or location, while the latter is a child frame of *'Being_located'* and focuses on the existence of some entity detected by some observer at certain location and time. We try to apply *'Existence'* frame with respect to something constantly present and we use frame *'Presence'* if something is present/relevant only at a certain time. However, often such things are difficult to judge, and our frame assignments may be disputable or interchangeable. Have a look at Example A.8.6. Actually, we could easily swap *'Existence'* and *'Presence'* labels in all of the utterances.

Example A.8.6. *Existence or Presence?*
*(a) [EXISTENCE Just make sure that **there's** a reference point somehow.]*
*(b) [EXISTENCE **There's** a graffiti saying JZ.]*
*(c) [EXISTENCE At one place **there are** two small spots.]*
*(d) [PRESENCE I could circle around the barrel and find out if **there's** a label on it.]*
*(e) [PRESENCE **There are** no signs of leakages around the area of the two barrels and the canister.]*
*(f) [PRESENCE **Are there** any further findings on that?]*

Other problematic pairs of frames are, for instance, *'Becoming_aware'* and *'Coming_to_believe'* or *'Required_event'* and *'Desirable_event'*. The frames here are not

connected by the *'see Also'* relation, but still in some contexts it is sometimes difficult to understand what is meant.

A.8.6 General or specific?

In Section A.6.1 we wrote that if some specific frame occurs only a couple of times, then we should prefer a more general one. Only if the nuances of meaning are relevant enough should we go with a more specific frame. However, sometimes it is not easy to decide whether those nuances are relevant or not. Moreover, there doesn't exist any strict rules telling us what is more important and what is less.

Have a look at Example A.8.7. Sentence *(a)* represents frame *'Containing'*. There are only two instances of this frame in our data. Both of them are evoked by the verb *'contain'*. Does this make them really different from instances of *'Locative_relation'* frame, which are mostly evoked by prepositions and adverbs?

Likewise, utterances *(b)* and *(c)* represent frames *'Be_piece_of'* and *'Be_subset_of'*, both of which occur only once in the whole corpus. The utterances describe different things, *(b)* is about fragments belonging to a partially destroyed building, and *(c)* is about power generator, which should be excluded from possible sources of fire. Are these differences important enough to keep two separate frames?

Utterance *(d)* is the only instance of frame *'Cause_of_temperature'* in the corpus. Should we keep it, or should we use *'Change_position_on_a_scale'* instead, which does not immediately relate to temperature?

Example A.8.7. *General or specific?*
(a) [$_{\text{CONTAINING}}$ *Yes, it's a blue canister standing upright and locked,* ***containing*** *Otalin.] (parent fame: 'Locative_relation')*
(b) [$_{\text{BE\_PIECE\_OF}}$ *I can also see fragments that* ***belong to*** *the building lying around here.] (parent frame: 'Being_included')*
(c) [$_{\text{BE\_SUBSET\_OF}}$ *That doesn't* ***count****.] (parent frame: 'Being_included')*
(d) [$_{\text{CHANGE\_OF\_TEMPERATURE}}$ *Can you find out if there's some heat* ***radiation****?] (parent frame: 'Change_position_on_a_scale')*

As Example A.8.7 shows, the aspect of specificity requires more careful thought, and probably some annotations need to be reconsidered.

A.9 Perspective

Another aspect of our data which may cause troubles is perspective. We distinguish the following interpretations of 'perspective'.

First interpretation has to do with the fact that we work with dialogues, i.e. an event may be viewed from two different points: either the speaker or the listener.

Compare utterances *(a)* and *(b)* in Example A.9.1. Both utterances have the verb *'try'* as a frame-evoking element. But the perspectives are different: while in *(a)* we have a team leader encouraging an operator to make a closer photo, in *(b)* the initiative comes from an operator themselves. Similarly, we use frames *'Waiting'* and

'Holding_ off_ on' to distinguish between own initiative/commitment and request, as utterances *(c)* and *(d)* illustrate.

Sometimes it can be difficult to tell what perspective is in focus. Sentences *(e)* and *(f)* demonstrate this. They get labels *'Imposing_ obligation'* and *'Being_ obligated'* respectively, however both sentences could be interpreted from opposite perspectives, so that the labels would be swapped.

Example A.9.1. *Two perspectives: speaker vs. listener*
*(a) [ATTEMPT_SUASION UGV1, please **try** to get results that are a little bit closer.]*
*(b) [ATTEMPT Yes, I can **try** that.]*
*(c) [WAITING I'll **wait** for another order.]*
*(d) [HOLDING_OFF_ON **Wait** a minute.]*
*(e) [IMPOSING_OBLIGATION Your **task** is to go to the selected area for you.]*
*(f) [BEING_OBLIGATED You don't **have to** go to the fire. Over.]*

Note that the FrameNet annotation guidelines [Ruppenhofer et al., 2006] do not separate *'Holding_ off_ on'* and *'Waiting'* frames based on perspective. The two frames are connected via a *'see Also'* relation, and no clear distinctions between them are given.

Second interpretation of perspective occurs due to the fact that an operator is often identified with a robot that they operate. As a result, most requests/questions concerting activities performed by robots sound as if they are performed by operators. In such cases two perspectives get merged into one. As illustration see utterances *(a), (c), (e)* and *(g)* in Example A.9.2. These utterances get the label *'Motion'* or *'Traversing'* despite the fact that the operator does not really move anywhere. It may seem that a more reasonable approach would be to assign the frame *'Operate_ vehicle'* to all of them. This would definitely work in sentences *(a)* and *(c)*. However, in other cases this would lead to a mismatch between the targets and the frames they evoke, as *'roll'*, *'climb'* or *'pass by'* have nothing to do with *'Operate_ vehicle'* frame. So, in order to be consistent, we treat an operator and their robot as a whole, and do not separate perspectives.

Actually, there are a few cases where an operator and their robot are clearly separated. Utterances *(b), (d)* and *(f)* in Example A.9.2 illustrate such a situation. In *(b)* and *(d* we have operator's perspective, and both utterances get *'Operate_ vehicle'* frame. Sentence *(f)* has clearly the perspective of the robot and gets a *'Traversing'* frame.

Example A.9.2. *Two perspectives: operator vs. robot*
*(a) [MOTION Can you **fly** around?]*
*(b) [OPERATE_VEHICLE Please, **fly** the drone over the top level of the left furnace and send a picture.]*
*(c) [MOTION Please, **drive** towards the smoke development.]*
*(d) [OPERATE_VEHICLE Please, **drive** back all the robots and then we'll meet here.]*
*(e) [MOTION Yes, I'm already **rolling** forwards.]*
*(f) [TRAVERSING UGV1 has now **climbed** upstairs.]*
*(g) [TRAVERSING That's where I should **pass by**, right?]*

A.10 Conclusion

So, above we summarized the key points of our approach to frame annotation of TRADR corpus, as well as some frequent issues that we faced with while annotating the data.

In conclusion we feel necessary to stress that while our annotation approach is based on the official FrameNet guidelines [Ruppenhofer et al., 2006], it also has its peculiarities due to the fact that TRADR data consists of dialogues and belongs to a rather specific domain. We focus more on predicates, especially verbal ones, and we are not so strict about targets being single tokens.

We tried to keep our annotations consistent, but still there exist some cases that need further consideration.

Appendix B

New Frames

Below we will present new frames that we had to introduce ourselves, as the current FrameNet database [FrameNet, 2020] does not contain any suitable candidates. We will give a frame label, a definition and examples for each frame. The definition of (non-)core frame elements is planned for future work.

Be_piece_of	
Inherits from:	Being_included
Definition:	A PART is considered to be a constituent of some entity described by the WHOLE. The relation is seen from the point of view of the PART.
Examples:	*I can also see [$_{PART}$ fragments] that **belong to** [$_{WHOLE}$ the building] [$_{PART}$ lying around here].*

Being_reasonable	
Inherits from:	Gradable_attributes
Definition:	Certain BEHAVIOR of PROTAGONIST is seen as practical and sensible.
Examples:	*As I can't see anything at the moment, it would definitely **make sense** if [$_{PROTAGONIST}$ you] [$_{BEHAVIOR}$ let the* UAV *guide you to some other points as soon as they've started again].*

Communication_by_protocol	
Inherits from:	Communication
Definition:	A COMMUNICATOR speaks to an ADDRESSEE using the phrases of special form (protocol) to establish/finish the conversation by radio.
Examples:	*[$_{COMMUNICATOR}$ Team leader] [$_{ADDRESSEE}$ for Tango].* *[$_{COMMUNICATOR}$ Team leader], **here** is [$_{ADDRESSEE}$ Tango].* *[$_{COMMUNICATOR}$* UAV*] [$_{ADDRESSEE}$ to* UGV-1*] please **answer**.* *[$_{COMMUNICATOR}$* UAV*] **speaking** [$_{ADDRESSEE}$* IDI*].*

Communication_fragment	
Inherits from:	None
Definition:	An auxiliary frame which serves the purpose of marking conversational fillers and sequences with unclear meaning. The frame is characterized by conflation of target and FRAGMENT itself.
Examples:	*[FRAGMENT Also... I'm with... erm...]* *[FRAGMENT Eeh eeh my my my...]* *[FRAGMENT Whether a person or its... below at the bottom edge there's a...]*

Communication_response_message	
Inherits from:	Statement
Definition:	A COMMUNICATOR gives a short usually positive or negative reply to an ADDRESSEE's question or request. Sometimes a TOPIC is also mentioned.
Examples:	***Roger*** *[TOPIC that], [ADDRESSEE team leader].* ***Okay***. ***Yes*** *[COMMUNICATOR by ground operator 1].*

Correction	
Inherits from:	Communication
Definition:	A COMMUNICATOR informs an ADDRESSEE that what the PATIENT has communicated is not right, true or suitable by providing the corrected version of the MESSAGE.
Examples:	*[COMMUNICATOR I] have to* ***correct*** *[PATIENT myself]: [MESSAGE* UGV-1*].*

Face_direction	
Inherits from:	State
Definition:	An ENTITY faces a particular DIRECTION.
Examples:	*For your information: [ENTITY it]'s* ***looking*** *[DIRECTION towards south].*

Lead	
Inherits from:	Cause_to_perceive
Definition:	An ENTITY leads in a particular DIRECTION or to some GOAL.
Examples:	*[ENTITY The stairwell]* ***leads*** *[DIRECTION upwards].* *There's smoke development at [ENTITY the first stairs] that* ***go*** *[DIRECTION upwards].*

Level_of_clarity	
Inherits from:	Gradable_attributes
Definition:	A DEGREE to which a REPRESENTATION is clear and detailed.
Examples:	*Yes, [$_{REPRESENTATION}$ the pictures] aren't [$_{DEGREE}$ very] **sharp**.*

Level_of_substance	
Inherits from:	Gradable_attributes
Definition:	A DEGREE of smoke in the air at some LOCATION.
Examples:	*It's actually [$_{DEGREE}$ quite] **smoky** [$_{LOCATION}$ DNI].*

Appendix C

Multiclass classification methods

In Chapter 2 we have already mentioned that one of the ways to systematize multiclass classification methods assumes splitting them into three groups: methods based on extension from binary classification, methods that assume transformation to multiple binary classifiers, and methods relying on hierarchical classification. In what follows we will present these three groups. This overview can also serve as a reference in case the reader needs additional information on how various frame-semantic parsers presented in Chapter 2 were trained.

All the algorithms we will consider are supervised, i.e. they work with labeled data. So, our training data has the form (x_i, y_i), where $x_i \in \mathbb{R}^n$ is the ith training instance and $y_i \in \{1, ..., K\}$ is the corresponding label. Under training a classifier we mean learning a model $\mathbb{H}$ so that $\mathbb{H}(x_i) = y_i$ for each new instance x_i [Aly, 2005]. If a classification problem is binary, usually $y_i \in \{0, 1\}$. Possible labels for multiple classes will be discussed below.

C.1 Extension from binary

In Chapter 2 we have discussed neural networks, which belong to this category, and mentioned some other approaches. We will briefly summarize them here.

C.1.1 Decision Trees

Decision tree is a classification algorithm that assumes building a tree, whose nodes are attributes, edges are attribute values, and leaves are classes. The tree is built by recursively splitting the training set, so that each new node is based on the attribute that brings the maximum information gain [Aly, 2005; Sun et al., 2009]. The tree can have as many leaves (classes) as necessary. A new sample is classified by following the path from the root to a leaf depending on the sample's attribute values.

Again, the standard version of the algorithm can be sensitive to the class imbalance problem. If some classes are rare, the algorithm needs to perform more splits to distinguish the classes. So, either the algorithm terminates before all necessary splits are done, or the leaves for minor classes get removed in the course of pruning as being susceptible to overfitting [Sun et al., 2009].

C.1.2 *k*-Nearest Neighbors

k-Nearest Neighbors is a very simple algorithm that does not actually require training a model. It works for any number of labels. It is assumed that all samples of a dataset lie in some common vector space. A new instance is classified by finding k nearest samples. The class is chosen by the majority vote. The number k is usually determined using a validation set, or by performing cross-validation. Various distance measures can be used, e.g., Euclidean distance or cosine similarity [Aly, 2005; Sun et al., 2009].

The algorithm is also prone to making errors when classifying instances of rare classes. As samples of such classes occur sparsely in the dataset, it is unlikely that the correct rare label will prevail over more frequent labels [Sun et al., 2009].

C.1.3 Naive Bayes

Naive Bayes classifier is also suitable for multiclass data. Given a new instance $x = (x_1, ..., x_N)$, we assign it a class c, such that $c = argmax_c P(C = c||x_1, ..., x_N)$, where $P(C = c||x_1, ..., x_N) = \frac{P(C=c)P(x_1,...,x_N||C=c)}{P(x_1,...,x_N)}$. In other words, we choose the most probable class given an instance, which is interpreted as a sequence of features.

To be able to apply the formula, we need to learn from the training set prior probabilities of classes $P(C_1), ..., P(C_K)$ and the probability of the given sequence given a certain class $P(x_1, ..., x_N||C = c)$. We see that the denominator is the same for all classes, so it can be discarded. If the sequence $x_1, ..., x_N$ is long enough, chances that it occurs in the training set are not high, and its conditional probabilities will likely be zero. To avoid this, the dependencies between the features are ignored, and the probability of the sequence given class is replaced with a product of probabilities of individual features: $P(x_1||C = c)...P(x_N||C = c)$ [Aly, 2005].

As shown in [Sun et al., 2009], Bayesian classifiers tend to learn patterns of dominant classes, and the patterns of small classes are hard to be encoded in the network. So, such classifiers are also prone to misclassifications given imbalanced data.

C.1.4 Support Vector Machines

The Support Vector Machines (SVM) classifier was originally designed only for binary classification problems. It is based on the following idea: data points lying in some common space can be separated by a hyperplane, so that the margin, i.e. the minimum distance from the hyperplane to the closest data points, is maximized. The data points that are the closest to the plane are called support vectors, hence the name of the classifier [Sun et al., 2009; Aly, 2005].

There exist extensions of the SVM classifier to multiclass versions. Most of them are based on the following idea: the classification task is treated as an optimization problem, and each class is represented by some new constraints and parameters added to this problem. Some information about multiclass SVM classifiers and related issues can be found, for instance, in [Aly, 2005].

SVM classifiers are considered to be robust and less prone to the data imbalance problem, if the class distribution is not too askew. This can be explained by the fact that the hyperplane separating the classes relies on only a few support vectors, and

the classes' sizes do not affect its position much [Sun et al., 2009]. On the other hand, Mahani and Ali [2019] write that in imbalanced datasets support vectors representing minority classes lie far from the hyperplane. As a result their contribution to the final decision is small, and minority classes get neglected. Mahani and Ali [2019] also discuss some algorithms that aim at overcoming this drawback.

C.2 Transformation to binary

A multiclass classification problem can be transformed into several binary classification problems. According to this approach, several binary classifiers are trained, and in order to determine the class of a new instance the outputs of all classifiers are compared [Aly, 2005]. Below we will present two the most widely used methods.

C.2.1 One-against-all

The One-against-all (OAA) method assumes training $N = K$ classifiers, where K is the number of classes. Each classifier in the ensemble treats the instances of a class in focus as positive, and the instances of all the other classes as negative, so it learns to discriminate between the target class and all the rest. Any suitable binary training algorithm can be used, e.g. SVM, decision tree or logistic regression [Machine Learning Mastery, 2020].

According to OAA approach, to avoid ambiguity each classifier outputs confidence scores rather than discrete labels. The decision is then based on the classifier that produces the maximum score [Wikipedia, 2020d].

Some researchers state that this method demonstrates performance that is comparable to more complex approaches, if the binary classifiers are tuned well [Aly, 2005]. However, there is a downside: large datasets and/or lots of classes require training a lot of models which may be costly from computational point of view [Machine Learning Mastery, 2020]. Moreover, OAA decomposition is sensitive to imbalanced data, and the resulting ensemble is prone to misclassifying instances of the minority classes [Lorena et al., 2008].

C.2.2 One-against-one

The One-against-one (OAO) approach is also an ensemble method. It is based on training $\frac{K(K-1)}{2}$ binary classifiers, each of which aims at distinguishing between two classes, and all the rest of the classes are ignored. Given a new instance, each classifier assigns a label to it, and the final decision is made via a majority vote [Machine Learning Mastery, 2020].

There are studies that show that OAO approach is better than OAA [Aly, 2005]. To the drawbacks of the method belong its tendency to overfitting if the dataset is small, and high possibility of contradictory voting [Esteves, 2020; Sun et al., 2009].

It is also necessary to mention that both OAA and OAO taken alone are not capable of integrating efficiently the information coming from separate classifiers [Esteves, 2020]. In order to overcome this drawback, as well as other shortcomings, different hybrid

techniques were developed. Some overviews of them can be found, for example in the works by Esteves [2020] or Lorena et al. [2008].

C.2.3 Error-Correcting Output-Coding

Error-Correcting Output-Coding (ECOC) is based on the idea of distributed output coding for neural networks that was described in Chapter 2. According to the ECOC approach, N binary classifiers are trained to differentiate between K classes. Each class is represented via a codeword of high and low bits (features) of length N (see Figure 2.2b), so that each separate classifier learns to recognize a certain feature. Given a new instance, all classifiers make their judgments, the output is compared with each of the K codewords via the *Hamming distance*, and the closest class is chosen Dietterich and Bakiri [1994].

It is important to mention that the length of a class codeword N should be bigger than the number of classes K. The additional bits introduce some redundancy in classes codification and allow the system to recover from possible classification errors made by separate classifiers. It is exactly why this approach has part 'error-correcting' in its name [Lorena et al., 2008].

C.2.4 Generalized coding

This approach was first developed by E. Allwein et al. [2000] and is a generalization of OAA, OAO and ECOC methods. Similar to the ECOC, each class label is represented by a codeword. However, in contrast to the ECOC, this codeword may contain three different values: 1 if a feature is present, -1 if it is absent, and 0 if the feature is not relevant for the given class. So, when each feature classifier is trained, all instances where this very feature is set to 0 are excluded from the training process. Given a new instance, it is classified like in the ECOC method.

It needs to be said that there exist other adaptations and modifications of the presented class decomposition techniques. An overview of them can be found, for instance, in [Lorena et al., 2008].

C.3 Hierarchical classification

Another way to handle multiclass data is to use hierarchical division of the output space, so that several binary classifiers form nodes of a binary tree or a more general structure like a directed acyclic graph (see Figure C.1). The leaves of the structure represent individual classes [Lorena et al., 2008].

The root classifier learns to perform a general discrimination first, and each child classifier performs a more refined partition until each leaf is associated with a certain class. In order to classify a new instance, one starts with the root classifier and based on its decision visits the next one. The procedure is repeated until a leaf is reached. There is no need to use all the classifiers - only those that form a certain path are engaged in the classification process.

One of the advantages of hierarchical classification is its lower complexity in comparison with the methods described above: the number of classifiers to be trained is much less than is required by, let's say, OAO approach. It is also necessary to point out an obvious disadvantage of the method. The performance of each classifier depends on how well its predecessors can differentiate classes. If errors are made, they accumulate, and in the result the net performance suffers [Lorena et al., 2008].

More information about different techniques within the approach is presented, for example, in works by Lorena et al. [2008] or Aly [2005].

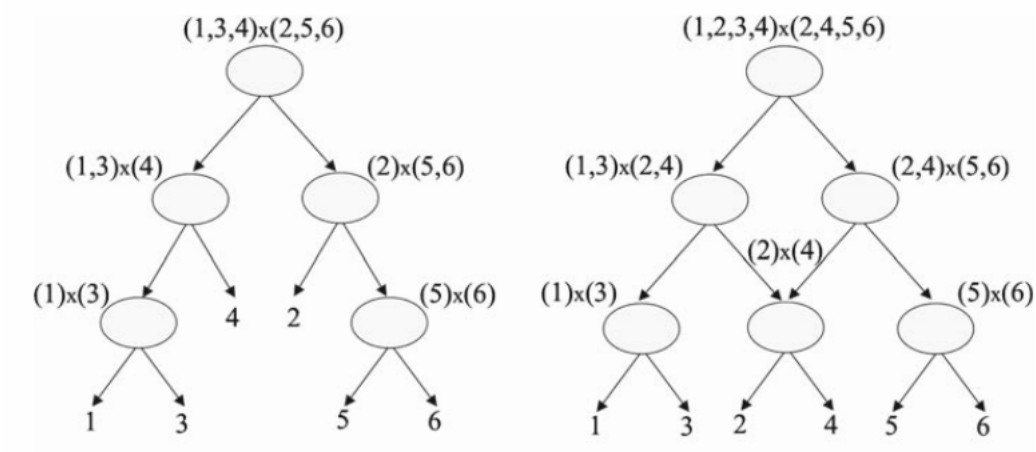


Figure C.1: Hierarchical classification for a problem with 6 classes: directed binary tree (left) and directed acyclic graph (right) [Lorena et al., 2008].

To conclude this survey, we would like to note that while earlier many frame-semantic parsers or frame classifiers relied on the transformation of the classification problem to binary classification tasks, nowadays, with the development of neural networks, the extension from binary to multiclass classification is more common.

Bibliography

Omri Abend and Ari Rappoport. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, 2013.

Tahira Alam, Chowdhury Farhan Ahmed, Sabit Anwar Zahin, Muhammad Asif Hossain Khan, and Maliha Tashfia Islam. An effective recursive technique for multi-class classification and regression for imbalanced data. *IEEE Access*, 7:127615–127630, 2019.

Jay Alammar. The Illustrated Transformer. `https://jalammar.github.io/illustrated-transformer/`, 2018. Accessed: 2020-10-17.

Waad Alhoshan, Riza Batista-Navarro, and L. Zhao. Towards a corpus of requirements documents enriched with semantic frame annotations. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 428–431, 2018.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1 (Dec):113–141, 2000.

Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19:1–9, 2005.

Tatiana Anikina and Ivana Kruijff-Korbayová. Dialogue act classification in team communication for robot assisted disaster response. In *Proceedings of SIGDIAL 2019*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Collin F. Baker. FrameNet, present and future. In *The First International Conference on Global Interoperability for Language Resources*, Hong Kong, 2008.

Collin F. Baker and Hiroaki Sato. The FrameNet data and software. In Yuji Matsumoto, editor, *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, page 161–164, 2003. URL `http://www.aclweb.org/anthology/P03-2030.pdf`.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada, 1998.

Collin F. Baker, Michael Ellsworth, and Katrin Erk. SemEval-2007 Task 19: Frame semantic structure extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, 2007.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186, 2013.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. Developing a large semantically annotated corpus. In *Eighth International Conference on Language Resources and Evaluation*, pages 3196–3200. European Language Resources Association (ELRA), 2012.

Chris Biemann. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City, June 2006. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W06-3812`.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, 2009.

Peter Bloem. Transformers from scratch. `http://peterbloem.nl/blog/transformers`, 2019. Accessed: 2020-10-20.

Paula Branco, Luis Torgo, and Rita Ribeiro. A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*, 2015.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168, 2002.

Harry Bunt. *Guidelines for using ISO standard 24617-2*. [s.n.], January 2019. TiCC TR 2019–1.

Aljoscha Burchardt, Anette Frank, and Manfred Pinkal. Building text meaning representations from contextually related frames - a case study. *Proceedings of IWCS-6*, page 188, 2005.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. The SALSA corpus: a German corpus resource for lexical semantics. In *LREC*, pages 969–974, 2006.

Miriam Butt, Helge Dyvik, Tracy King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. 2004. doi: 10.3115/1118783.1118786.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1): 321–357, June 2002. ISSN 1076-9757.

Dawei Chen and Jieyue He. CBOS-clustering base on the score for motif discovery in biological network. In *2014 Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, pages 845–850, 2014. doi: 10.1109/IMCCC.2014.178.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An analysis of BERT's attention, 2019.

Ann Copestake. Invited talk: Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, 2009.

Bonaventura Coppola, Alessandro Moschitti, Sara Tonelli, and Giuseppe Riccardi. Automatic Framenet-based annotation of conversational speech. In *2008 IEEE Spoken Language Technology Workshop*, pages 73–76, 2008.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 948–956, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N10-1138`.

Tom De Smedt and Walter Daelemans. Pattern for Python. *The Journal of Machine Learning Research*, 13(1):2063–2067, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N19-1423`.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

Disaster Robotics Research Project. Long-term human-robot teaming for robot-assisted disaster response (TRADR). `http://www.tradr-project.eu/`, 2020. Accessed: 2020-04-30.

Vitor Miguel Saraiva Esteves. Techniques to deal with imbalanced data in multi-class problems: A review of existing methods, 2020.

Charles J. Fillmore. The case for case. The Ohio State University, 1967.

Charles J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976. doi: 10.1111/j.1749-6632.1976.tb25467.x. URL `https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-6632.1976.tb25467.x`.

Charles J. Fillmore. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea, 1982.

Charles J. Fillmore and Collin F. Baker. Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*, volume 6, 2001.

FrameNet. The official website for the FrameNet project. `https://framenet.icsi.berkeley.edu/fndrupal/`, 2020. Accessed: 2020-04-30.

Vicente García, Ramón Alberto Mollineda, and José Salvador Sánchez. Index of balanced accuracy: A performance measure for skewed class distributions. In *Iberian conference on pattern recognition and image analysis*, pages 441–448. Springer, 2009.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

Alex Harlan. You might be leaking data even if you cross validate. `https://alexforrest.github.io/you-might-be-leaking-data-even-if-you-cross-validate.html`, 2020. Accessed: 2021-01-20.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. Out-of-domain FrameNet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 471–482, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-1045`.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1458, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1136. URL `https://www.aclweb.org/anthology/P14-1136`.

Julia Hockenmaier and Mark Steedman. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, 2007.

Matthew Honnibal and Ines Montani. SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. Do attention heads in BERT track syntactic dependencies?, 2019.

Nancy Ide. The American National Corpus: Then, now, and tomorrow. In *Selected Proceedings of the 2008 HCSNet Workshop on Designing the Australian National Corpus: Mustering Languages, Summerville, MA. Cascadilla Proceedings Project*, 2008.

Nancy Ide. Case study: The Manually Annotated Sub-Corpus. In *Handbook of Linguistic Annotation*, pages 497–519. Springer, 2017.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Richard Johansson and Pierre Nugues. LTH: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 227–230, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S07-1048`.

Alexandre Kabbach and Corentin Ribeyre. Valencer: an API to query valence patterns in FrameNet. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 156–160, 2016.

Aditya Kalyanpur, Or Biran, Tom Breloff, Jennifer Chu-Carroll, Ariel Diertani, Owen Rambow, and Mark Sammons. Open-domain frame semantic parsing using transformers, 2020.

Aishwarya Kamath and Rajarshi Das. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*, 2018.

Hans Kamp, Josef Van Genabith, and Uwe Reyle. Discourse representation theory. In *Handbook of philosophical logic*, pages 125–394. Springer, 2011.

Rohit J. Kate and Yuk Wah Wong. Semantic parsing. The task, the state of the art and the future. In *Tutorial abstracts of the 20th Meeting of the Association for Computational Linguistics*, page 6, 2010.

Alexander Koller, Stephan Oepen, and Weiwei Sun. Graph-based meaning representations: Design and processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 6–11, 2019.

Michał Koziarski. Radial-based undersampling for imbalanced data classification. *Pattern Recognition*, 102:107262, 2020. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107262. URL `http://www.sciencedirect.com/science/article/pii/S0031320320300674`.

Ivana Kruijff-Korbayová, Francis Colas, Mario Gianni, Fiora Pirri, Joachim de Greeff, Koen Hindriks, Mark Neerincx, Petter Ögren, Tomáš Svoboda, and Rainer Worst. TRADR project: Long-term human-robot teaming for robot assisted disaster response. *KI - Künstliche Intelligenz*, 29(2):193–201, Jun 2015. ISSN 1610-1987. doi: 10.1007/s13218-015-0352-5. URL `https://doi.org/10.1007/s13218-015-0352-5`.

Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Citeseer, 1997.

Keita Kurita. An overview of normalization methods in deep learning. https://mlexplained.com/2018/11/30/an-overview-of-normalization-methods-in-deep-learning/, 2018a. Accessed: 2020-10-24.

Keita Kurita. Weight normalization and layer normalization explained (Normalization in deep learning Part 2). https://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/, 2018b. Accessed: 2020-10-24.

Carole Lailler, Anaïs Landeau, Frédéric Béchet, Yannick Estève, and Paul Deléglise. Enhancing the RATP-DECODA corpus with linguistic annotations for performing a large range of NLP tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1047–1050, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1166.

Egoitz Laparra and German Rigau. Exploiting explicit annotations and semantic types for implicit argument resolution. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 75–78. IEEE, 2012.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.

Chahira Lhioui, Anis Zouaghi, and Mounir Zrigui. A rule-based semantic frame annotation of Arabic speech turns for automatic dialogue analysis. *Procedia Computer Science*, 117:46–54, 2017.

Christine A. Lindberg and Angus Stevenson. *New Oxford American Dictionary*. Oxford University Press, 1999.

Ana Carolina Lorena, André C.P.L.F. De Carvalho, and João M.P. Gama. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30(1-4):19, 2008.

Machine Learning Mastery. One-vs-rest and one-vs-one for multi-class classification. https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/, 2020. Accessed: 2020-09-23.

Aouatef Mahani and Ahmed Riad Baba Ali. Classification problem in imbalanced datasets. In *Recent Trends in Computational Intelligence*. IntechOpen, 2019.

Wing W.Y. Ng, Junjie Hu, Daniel S. Yeung, Shaohua Yin, and Fabio Roli. Diversified sensitivity-based undersampling for imbalance classification problems. *IEEE transactions on cybernetics*, 45(11):2402–2412, 2014.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. MRP

2019: Cross-framework meaning representation parsing. In *CoNLL Shared Task*, pages 1–27, 2019.

Rebecca J. Passonneau, Collin Baker, Christiane Fellbaum, and Nancy Ide. The MASC word sense sentence corpus. In Mehmet Ugur Dogan, Joseph Mariani, Asuncion Moreno, Sara Goggi, Khalid Choukri, Nicoletta Calzolari, Jan Odijk, Thierry Declerck, Bente Maegaard, Stelios Piperidis, Helene Mazo, and Olivier Hamon, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC 2012*, Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC 2012, pages 3025–3030. European Language Resources Association (ELRA), jan 2012. 8th International Conference on Language Resources and Evaluation, LREC 2012 ; Conference date: 21-05-2012 Through 27-05-2012.

Douglas B. Paul and Janet Baker. The design for the Wall Street Journal-based CSR corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

Miriam R.L. Petruck. *Frame Semantics*. John Benjamins, 1996.

Ronaldo C. Prati, Gustavo E.A.P.A. Batista, and Maria Carolina Monard. Class imbalances versus class overlapping: An analysis of a learning system behavior. In *Mexican international conference on artificial intelligence*, pages 312–321. Springer, 2004.

Patti Price. Evaluation of spoken language systems: The ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

SALSA Project. Software. `http://www.coli.uni-saarland.de/projects/salsa/page.php?id=software`, 2020. Accessed: 2020-12-16.

Kamthorn Puntumapon, Thanawin Rakthamamon, and Kitsana Waiyamai. Cluster-based minority over-sampling for imbalanced datasets. *IEICE TRANSACTIONS on Information and Systems*, 99(12):3101–3109, 2016.

Christian Raymond, Kepa Joseba Rodriguez, and Giuseppe Riccardi. Active Annotation in the LUNA Italian Corpus of Spontaneous Dialogues. In *LREC*, 2008.

Eugénio Ribeiro, Andreia Sofia Teixeira, Ricardo Ribeiro, and David Martins de Matos. Semantic frame induction as a community detection problem. pages 274–285, 2020.

Michael Roth and Anette Frank. Inducing implicit arguments from comparable texts: A framework and its applications. *Computational Linguistics*, 41(4):625–664, 2015.

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California, 2006. Distributed with the FrameNet data.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin F. Baker, and Martha Palmer. SemEval-2010 task 10: Linking events and their participants in discourse. 2010.

Alexander Rush, Vincent Nguyen, and Guillaume Klein. The Annotated Transformer. `http://nlp.seas.harvard.edu/2018/04/03/attention.html`, 2018. Accessed: 2020-10-17.

Sabyasachi Sahoo. Residual blocks - building blocks of ResNet. `https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec`, 2018. Accessed: 2020-10-26.

Nathan Schneider and Chuck Wooters. The NLTK FrameNet API: Designing for discoverability with a rich linguistic resource. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 1–6, 2017.

Leonid A. Sevastianov and Eugene Yu Shchetininb. On methods for improving the accuracy of multi-class classification on imbalanced data. *marketing*, 2:3, 2020.

Jennifer Sikos and Sebastian Padó. Frame identification as categorization: Exemplars vs prototypes in Embeddingland. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 295–306, Gothenburg, Sweden, May 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-0425. URL `https://www.aclweb.org/anthology/W19-0425`.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

Yanmin Sun, Andrew K.C. Wong, and Mohamed S. Kamel. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04):687–719, 2009.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. Syntax-infused transformer and BERT models for machine translation and natural language understanding, 2019.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *ArXiv*, abs/1706.09528, 2017.

Sang-Sang Tan and Jin-Cheon Na. Positional attention-based frame identification with BERT: A deep learning approach to target disambiguation and semantic frame selection, 2019.

Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2020.

Sara Tonelli and Rodolfo Delmonte. VENSES++: Adapting a deep semantic processing system to the identification of null instantiations. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 296–299, 2010.

Jeremy Trione, Frederic Bechet, Benoit Favre, and Alexis Nasr. Rapid FrameNet annotation of spoken conversation transcripts. In *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11)*, London, UK, April 2015. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W15-0212`.

Vincent Van Asch. Macro-and micro-averaged evaluation measures (basic draft). *Belgium: CLiPS*, 49, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Jesse Vig. Deconstructing BERT, Part 2: Visualizing the inner workings of attention. `https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1`, 2019. Accessed: 2020-10-20.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding, 2019.

Wikipedia. American National Corpus. `https://en.wikipedia.org/wiki/American_National_Corpus`, 2020a. Accessed: 2020-08-27.

Wikipedia. British National Corpus. `https://en.wikipedia.org/wiki/British_National_Corpus`, 2020b. Accessed: 2020-08-27.

Wikipedia. Geometric mean. `https://en.wikipedia.org/wiki/Geometric_mean`, 2020c. Accessed: 2020-10-01.

Wikipedia. Multiclass classification. `https://en.wikipedia.org/wiki/Multiclass_classification`, 2020d. Accessed: 2020-09-23.

Wikipedia. Nuclear Threat Initiative. `https://en.wikipedia.org/wiki/Nuclear_Threat_Initiative`, 2020e. Accessed: 2020-08-27.

Wikipedia. One-hot. `https://en.wikipedia.org/wiki/One-hot`, 2020f. Accessed: 2020-09-14.

Christian Willms, Constantin Houy, Jana-Rebecca Rehse, Peter Fettke, and Ivana Kruijff-Korbayová. Team communication processing and process analytics for supporting robot-assisted emergency response. In *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 216–221. IEEE, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art natural language processing, 2020.

William Woods. The lunar sciences natural language information system, 1972.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization, 2019.

Bishan Yang and Tom Mitchell. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, 2017.

Hongyan Zhao, Ru Li, Fei Duan, Zepeng Wu, and Shaoru Guo. TSABCNN: Two-stage attention-based convolutional neural network for frame identification. In Maosong Sun, Ting Liu, Xiaojie Wang, Zhiyuan Liu, and Yang Liu, editors, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 289–301, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01716-3.

Qile Zhu, Xiyao Ma, and Xiaolin Li. Statistical learning for semantic parsing: A survey. *Big Data Mining and Analytics*, 2(4):217–239, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.