CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Automatic Semantic Role Labelling (SRL) in Swedish

Transformer-based Swedish SRL Through Transfer Learning

Master's thesis in Applied Data Science, Computer science and engineering

Lucy YANG BUHR

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Automatic Semantic Role Labelling (SRL) in Swedish

Transformer-based Swedish SRL Through Transfer Learning

Lucy YANG BUHR

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Automatic Semantic Role Labelling (SRL) in Swedish
Transformer-based Swedish SRL Through Transfer Learning
Lucy Yang Buhr

Supervisor at GU: Dana Dannélls
Examiner at CSE: Richard Johansson

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (<To Be Added>)

Typeset in LaTeX
Gothenburg, Sweden 2023

Automatic Semantic Role Labelling (SRL) in Swedish
Transformer-based Swedish SRL Through Transfer Learning
Lucy Yang Buhr
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this paper, with using advanced deep learning networks, we built the first end-to-end semantic role labelling model (SRL) for Swedish texts. This Swedish SRL model can, with a given Swedish sentence, perform trigger identification, frame classification and argument extraction tasks automatically in one shot. Using the semantic annotation examples from the Swedish FrameNet corpus (SweFN), we experiment with two transfer learning approaches. We show that the Swedish SRL based on a pre-trained English SRL can speed up the training, meanwhile the multilingual model (mT5) based model has better performance when the possible frames are unknown. Through extensive empirical analysis of the model performance, we point out the major factors that can further improve the results.

Keywords: Semantic role labelling, SRL, transfer learning, multi-task learning, multilingual transfer, T5, mT5

Acknowledgements

We thank our supervisor, Dana Dannélls, for helping us to understand the linguistic concepts and FrameNet corpus, for her full supports, careful review and helpful comments. We thank our examiner, Richard Johansson, for his inspiring discussions and valuable feedbacks. We are grateful to David Chanin for sharing his pre-trained English SRL model and facilitating us to transfer learn on this model. We deeply appreciate all the supports received from our family during the whole project period.

Lucy Yang Buhr, Gothenburg, 2023-06-01

Contents

1

Introduction

1.1 The problem

Natural Language Processing (NLP) is a field of artificial intelligence (AI) that focuses on the interaction between computers and human language. It involves a range of techniques, such as statistical analysis, machine learning, and deep learning, to process, analyze and generate natural language data. Some of the key tasks in NLP include text classification (categorizing text into predefined categories), named entity recognition (NER, identifying and classifying entities in a text), machine translation, and more.

In NLP, *semantic role labelling* (SRL) is an automatic natural language understanding (NLU) technique that parses semantic arguments with phrases or words in a sentence. These semantic arguments, typically centralized around a word (also called "predicate") or a phrase , provide essential details of the sentence to answer questions such as "who?", "did what?", "to whom?", "how?", "when?", and "where?" [1]. Predicate-arguments express the semantic roles of words or phrases in a sentence, and the granularity of these semantic roles can vary, depending on the semantic theory and lexical resources being used (see Section 2.1).

Structured predicate-arguments can help a machine to better "understand" the core meaning of a text. This can be illustrated with the following examples for the verb "break" ([1], [2]):

- John[AGENT] broke the window[THEME] with a rock[INSTRUMENT].
- The rock[INSTRUMENT] broke the window[THEME].
- The window[THEME] was broken by John[AGENT].

In spite of the different appearances of the above three sentences, same words or phrases play the same semantic roles around "broke". The fact that semantic roles can be identified regardless of their location in the sentence assists a NLP network to extract the central meaning of the text. Therefore, the SRL technique is valuable for natural language understanding and for developing NLP applications, such as information extraction ([3], [4]), question answering [5], machine translation [6], and even modern voice assistants [7].

Traditionally, when using statistical methods, developing SRL systems required syn-

tactic analysis of the sentences ([8], [9], [10]). In recent years however, thanks for the breakthroughs of deep neural networks, such as LSTM-RNNs[1] and transformers (see Section 2.3), SRL modelling can be formed as end-to-end deep learning models with automatic labelling more detail-categorized semantic roles under a specific situation (called "frames", see Section 2.1).

However, most of these advanced SRL models, that employ deep learning techniques, are trained for English. For Swedish, the only published study on developing a SRL [12] model was done in 2012, before the era of deep learning algorithms, when the Swedish FrameNet (SweFN, see Section 2.1) started. Since then, there has not been much advances in automate semantic role labelling for Swedish texts. Hence, no established Swedish SRL model that utilizing the state-of-the-art techniques.

For English, the existing transformer-based SRL models are often developed with supervised training, which means we need semantic role annotated sentences for training. For state-of-the-art (SOTA) techniques, such as transformer, the annotated dataset has to be very big. For Swedish, the available lexical resource SweFN, although being further developed during the past ten years, is still small comparing to the English correspondence.

Lack of data makes transfer learning (TL, see Section 2.5) a most desirable approach when developing a Swedish SRL model with SweFN. It makes it possible to re-use the "knowledge" (the relationship between words or sub-words) already learned from previous training, thus no need of training from scratch. This, in turn, reduces the demand of training data and improve training efficiency.

Therefore in this study, we aim to develop an end-to-end Swedish SRL by SOTA techniques and transfer learning on two publicly available deep learning networks:

- Frame-semantic-transformer[2]: a ready trained English SRL (see Section 2.6.2) based on a transformer system, T5 (see Section 2.6.1);
- mT5: a pre-trained multilingual transformer based architecture (see Section 2.6.3).

1.2 Aims

The general aim of this project is to set up an automatic SRL model for Swedish with a transformer architecture through transfer learning and supervised training on the annotated examples from SweFN. In particular we will focus on following problems:

(A) Given the constraints of current dataset SweFN, how should we develop a Swedish SRL model through transfer learning? This question is going to be explored through two types of TL techniques.

[1] More about LSTM-RNNs: refer to paper [11]

[2] `https://pypi.org/project/frame-semantic-transformer`

(B) Which part of the dataset suits better for the training? For this question, we will analyze the model performance for each semantic frame (see Section 4.3).

1.3 Ethical considerations

One potential ethical issue concerning this project is *idea plagiarism*. As this study involves re-using established training procedures from other studies, failure of accrediting to the original author can be seen as acting unfairly to gain undeserved advantage in a professional competition.

On the other side, there is no authorization or privacy issue. Because the Swedish SRL will be trained on the published open-domain dataset, SweFN. And the transfer learning models are also publicly available.

In term of computational costs, thanks for leveraging transfer learning existing large-language-models, none of our Swedish SRL models are developed from scratch without large-scale pre-training, which leads to comparably low energy consumption in this study.

2

Theory and Background

2.1 FrameNet and SweFN

For English, there are mainly two types of lexical resources that can facilitate the supervised training of semantic role labelling models: The FrameNet resource (introduced in 1998 by Baker et al., [13]) and the PropBank resource (published in 2002 by Kingsbury and Palmer [14]). Both are publicly available and contain large text corpora of semantic annotations with example sentences done by lexicographers and linguists. The biggest difference between PropBank and FrameNet lies in the fact that PropBank is a verb-oriented in creating predict-argument structure and FrameNet is frame-based that groups similar words into a semantic frame. A frame is a semantic representation of a situation, an object or an event and can be evoked by predefined set of words, lexical units (LU)[1]. LUs can be verbs, nouns or other types of part-of-speech entities.

Since a word may have different meanings, it can also have various semantic roles under different *semantic frames*. In case of multiple meanings, the same word will be annotated as different LUs under respective frames. For instance, the word "play" evokes frame PERFORMERS-AND-ROLES when using in sentence: "Margaret PLAYED Juliet in this film", and also evokes frame COMPETITION when using in sentence: "He and I PLAYED tennis".

In Berkeley FrameNet (BFN), each frame distinguishes the associated semantic roles between core frame elements (FEs) and non-core frame elements. The core FEs are essential to the specific frame while non-core FEs generally add more details about the event of the frame (such as manner, time, place). Figure 2.1 shows the definition of semantic frame "APPLY HEAT". Here COOK, FOOD, TEMPERATURE-SETTING and HEATING-INSTRUMENT belong to core FEs, while DURATION, MEDIUM and CO-PARTICIPANT are non-core FEs.

Definition:

A Cook applies heat to Food, where the Temperature_setting of the heat and Duration of application may be specified. A Heating_instrument, generally indicated by a locative phrase, may also be expressed. Some cooking methods involve the use of a Medium (e.g. milk or water) by which heat is transferred to the Food. A less semantically prominent Food or Cook is marked Co-participant.

Figure 2.1: FrameNet definition of the semantic roles under semantic frame "APPLY HEAT". *Source: FrameNet.*

[1] More description about LU: `https://www.nltk.org/howto/framenet.html`

BFN also provides annotated example sentences of each LU under the defined semantic frames to illustrate several possible semantic structures around the LU. For example, under the frame of APPLY HEAT, the sentence "*Ellen FRIED the eggs with chopped tomatoes and garlic.*" is annotated in the following way:

- *FRIED* is the LU, the trigger evoking frame APPLY HEAT.
- *Ellen* has the semantic role of COOK.
- *the eggs* has the role of FOOD.
- *with chopped tomatoes and garlic* has the role of CO-PARTICIPANT .

With increasing applications of Berkeley FrameNet in training NLP models, FrameNet has been developed for many other languages globally, such as German, Swedish, French, Brazilian Portuguese, Spanish, and also Japanese, Chinese, and Korean.[2] Under the linguistic hypothesis that semantic frames are language independent to a large degree, FrameNets for these non-English languages are generally created by following the structure of the semantic frames defined in BFN. For example, the frame names, as well as FEs, and the relations between the frames, are transferred from the English FrameNet with some modifications([15], chapter 8).

The Swedish FrameNet (SweFN)[3] was started in 2012 within the project SweFN^{++} [15] and has been further developed over the past decade. Following the structure of BFN, SweFN is also frame-orientated. The SweFN team has focused on re-using and expanding the BFN-defined frames and the list of frame elements contained in these frames. Thus, both frames and FEs remain in English.

In SweFN, the lexical units that evoke these frames are all linked to a Swedish dictionary, SALDO ([15], chapter 3). Since each SALDO entry can only correspond to no more than one semantic frame, when a word evokes several semantic frames, each meaning has to be defined as a different LU.

Although SweFN is regarded as a Swedish version of Berkeley FrameNet, the semantic annotation structures of these two databases are still different from each other in multiple ways. Moreover, there are frames in the BFN that do not exist in the SweFN and vise versa. Some frames have been modified and do not contain the same semantic roles[4].

Furthermore, Berkeley FrameNet contains both fully annotated sentences (approx. 6,000 [16]) and single frame annotated sentences (called "examplars"). SweFN, at current stage, has only annotations of single frame per sentence. It is because every example sentence in SweFN is independently annotated with focus on one semantic frame. But if a sentence have several triggers that can evoke the same frame in focus, all the triggers will be annotated as LUs. Actually, 5% sentences in the SwedFN dataset are in this case.

[2]Source: `https://www.globalframenet.org`

[3]Source: `http://spraakbanken.gu.se/eng/swefn`

[4]`https://spraakbanken.gu.se/projekt/swedish-framenet-swefn/documentation-for-swefn`

This single-frame annotation can be a drawback for identification of possible triggers for multiple frames (see Section 3.4.1). Because base on current dataset and the chosen implementation, the Swedish SRL model trained in this study can only tag one trigger in a given sentence, even if there could have been multiple frames.

	Semantic Frames	Total Lexical Units	Example Sentences
SweFN	1 195	39 212	9 K
BFN	1 221	13 572	202 K

Table 2.1: SweFN statistics compared to Berkeley FrameNet 1.7 (BFN)

There are several versions of BFN. The latest release is Berkeley FrameNet 1.7 (published in 2015), which contains more than 200K annotated examples (approx. 20 sentences per LU). As shown in Table 2.1, at the present there are much fewer annotated sentences in SweFN compared to BFN. In the Swedish FrameNet, the annotated examples do not cover all LUs or frames. In addition, the available examples are not distributed evenly among frames (shown in Figure 2.2), 860 out of 1195 frames contain less than 10 example sentences. The frame "EMPTYING" has the highest amount of annotated sentences, in total 67 sentences.

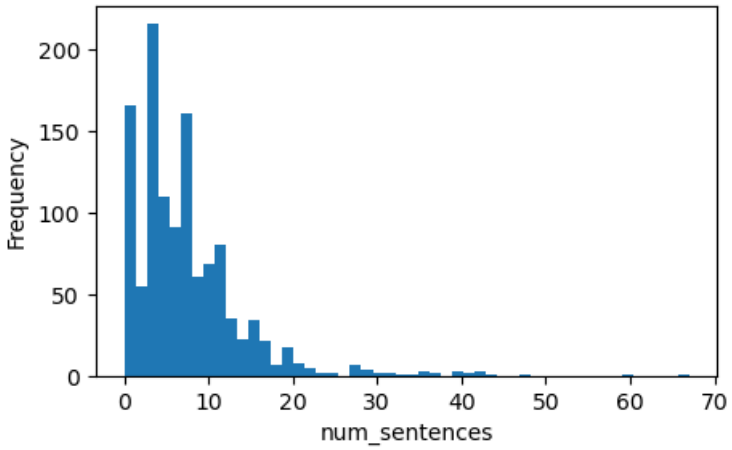

Figure 2.2: Distribution of number annotated sentences per frame in SweFN.

Fewer annotated examples and more triggers of the data resource SweFN means it will be more challenging to train a SRL model. Furthermore, the extremely unbalanced dataset distribution among semantic frames can also mean that it is not likely to expect the Swedish SRL developed with SweFN to perform equally well among all frames.

2.2 Semantic Role Labelling (SRL)

SRL, also called as "frame semantic parsing", is to automatically annotate according to the frame and its semantic roles defined in a FrameNet. That is, given a sentence

as the input, e.g. "*Ellen fried the eggs with chopped tomatoes and garlic.*", the output of a frame semantic parser should be as following:

1. *Trigger Identification*: allocating "*FRIED*" as the trigger.
2. *Frame Classification*: classifying the triggered frame APPLY HEAT.
3. *Argument Extraction*: identifying the locations of the three FEs spans under this triggered frame, and labelling these spans with their respective semantic roles: COOK, FOOD and CO-PARTICIPANT.

Very often, a frame semantic parser covers only the last one or two sub-tasks (e.g. [17], [18], [19], [20]), with the trigger (or also the targeted frame) already provided by the user. But, when using a encoder-decoder transformer (see Section 2.4) such as T5, frame semantic parsing can be easily handled in a pipeline of all the three sub-tasks ([16], [21]).

2.3 SRL modelling

SRL is commonly trained through supervised learning on a large frame semantic dataset. It was introduced in a computational linguistic study in 2002 by Gildea et al. and was based on statistical classifiers trained on the annotated sentences from the FrameNet [17] with using syntactic properties and grammatical characteristics. Since the release of PropBank corpus, more machine learning models have been developed through manual feature engineering approaches ([22], [23], [24]) that focused on utilizing the lexicon-syntactic features.

More recent SRL models have been developed using syntax-agnostic deep neural networks, which allows to train a semantic role labelling model in an end-to-end fashion and directly use input embedding without consideration of syntactic features [25]. Since then, there are more works that explore recurrent neural network (RNN), such as long short-term memory (LSTM), in developing SRL models ([8], [9], [26]). These deep neural networks had reached, at the time, state-of-the-art (SOTA) performances for English frame semantic parsing.

NLP has further advanced since the publishing of transformer architecture (see Section 2.4) and its variants. Several studies have demonstrated the usage of transformer and self-attention technique (see Section 2.4.1) being able to leverage multi-task learning in a SRL ([27], [28]). It has been shown that transformer-based models trained on a large semantic annotation corpus, e.g. FrameNet, can outperform earlier SOTA semantic role labelling approaches ([18], [29], [30]).

Since the introduction of pre-trained large language models, transfer-learning has become popular. Recently, through using transformer architecture and its variants, Kalyanpur et al. have explored frame semantic parsing [18] techniques with training on BFN dataset. In this study, several pre-trained transformers-based models have been compared, such as encoder-decoder T5 and language generative model GPT-2 [31]. This research shows that the pre-trained T5 system is able to improve the SOTA benchmark by 12-17% in prediction accuracy.

In 2021, Oliveira et al. from University of Porto investigated the improvement of Portuguese semantic role labelling with transformers and transfer learning [32]. Their experiment demonstrated that through transfer learning of a BERT-based model (see Section 2.4) with adding a few layers, the new SRL model can suppress the previous state-of-the-art in Portuguese SRL by over 15 F1 points in prediction accuracy. While on the other side, the labelling here is based on a rather coarse identified semantic roles, i.e. "Who did What to Whom, How, When and Where" [32], thus only six roles to identify, which is less than half of the size defined in FrameNet or SweFN (more than 12 semantic roles per frame on average). It is worth noting that this Portuguese SRL model performing only the third sub-task (see Section 2.2.), identification and classification of FEs, with pre-defined trigger and frame name. In contrast, the automatic Swedish SRL in this work will target all the sub-tasks, which is more challenging.

For Swedish there had been some work previously done on semantic role labelling using SweFN [12] in 2012, during the developing of SweFN. Due to being prior to modern deep neural network and transformer, this study is conducted using various feature engineering approaches and syntactic parsing, with traditional machine learning techniques. In this thesis work, we investigate Swedish semantic role labelling derived from transformer models, SOTA deep learning techniques.

2.4 Transformer

The Transformer is a deep learning model architecture that was introduced in a 2017 paper by Vaswani et al [33]. The core idea of the transformer is solely relying on the concept of *self-attention*, which takes a set of queries, keys, and values as input and computes a set of attention scores for each element in the input sequence. These attention scores are then used to compute a weighted sum of the values, which is the final output of the self-attention layer. Through this, the transformer is able to selectively focus on most strongly related items of the input sequence depending on the context. This enables the model to capture long-range dependencies between the elements in the input sequence more effectively than previous sequence-to-sequence models like the recurrent neural network (RNN).

In recent years, transformer-based models have made significant advancements in the field of in natural language processing. The models, such as BERT[5], GPT[6], T5 (see Section 2.5) and their variants, have established a new standard in language inference and generation tasks. BERT was published by Google in 2019 as a pre-trained transformer encoder, and focuses on feature extraction from a text. Both BERT and T5 are common choices for developing modern semantic role labelling models, but with BERT, it requires using different heads on the top for different sub-tasks.

[5]BERT stands for "Bidirectional Encoder Representations from Transformers", published in paper [34]

[6]GPT stands for "Generative Pre-trained Transformer", introduced in paper [35]

2.4.1 Self-attention

Self-attention is a useful attention mechanism that can connect together each position of a sequence when generating the representation of that sequence. In a self-attention mechanism, the input is first transformed into three parts, commonly referred to as queries Q, keys K, and values V. These parts are typically obtained from the same input X using linear transformations with query-, key- and value-weight matrices. Thus, $Q = W_Q X$, $K = W_K X$, $V = W_V X$.

Then, the model computes a similarity score between each query and key. This is done using a dot product $(Q \cdot K^T)$ and a scaling factor $\frac{1}{\sqrt{d_k}}$ (the dimension of keys), followed by a *softmax* function $(exp(z_i)/exp(z).sum)$ to convert the scores into weights W_{si} that represent the importance of each key for each query q_i.

Finally, for each input vector position i, the self-attention output y_i for each query Q_i is weighted combination of the value at that position V_i. That is $y_i = \sum_{s=1}^{n} W_{si} \cdot V_i$. Since the weights reflect the similarity or dependency between the query and the corresponding key, self-attention can help the model focus on important information and relationships within the input data. This can therefore help improve the performance of the model on a wide range of natural language processing tasks.

2.4.2 Transformer architecture

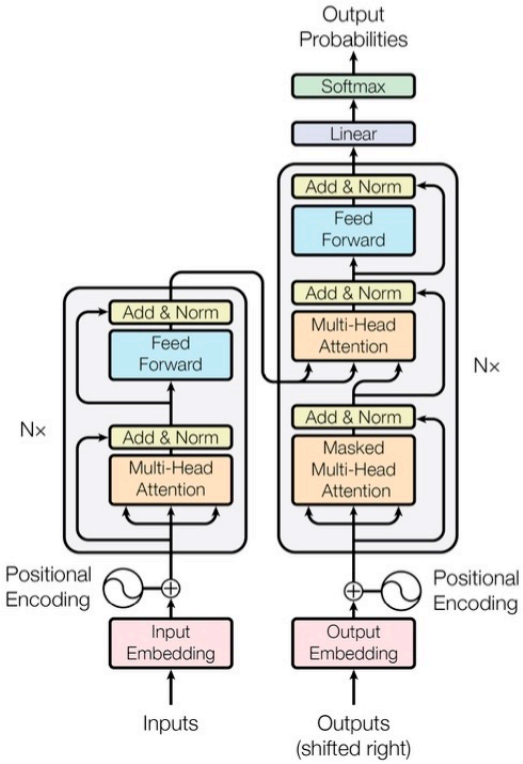

Figure 2.3: Standard transformer contains encoder network (left part) and decoder network (right part). "Add & Norm" means *residual connection* and *layer normalization*. *Source*: [33].

In a transformer model there are two sub-models: the encoder and the decoder, shown in Figure 2.3. The encoder takes the input sequence and produces a set of hidden representations, while the decoder uses these representations to generate the output sequence. Both encoder and decoder is constructed of **N** identical layers.

Each encoder layer has a multi-head self-attention sub-layer, and a sub-layer with fully connected feed-forward network (FFN). The multi-head attention performs self-attention multiple times with different sets of queries, keys, and values to capture different aspects of the input sequence. The FFN uses one hidden layer to perform two linear transformations, by using a $ReLU$ activation function (i.e. $max(0, x)$) after the first transformation, thus $max(0, xW_1 + b1)W_2 + b_2$.

Each decoder layer contains three sub-layers: multi-head self-attention, multi-head "encoder-decoder attention", and a feed-forward network. Here, the FFN is identical as the one in encoder, while the multi-head self-attention in decoder allows each position attending only up to and including that position and mask out all following "un-seen" positions. And the "encoder-decoder attention" takes queries from the previous decoder multi-head self-attention layer, and the keys and values are the output of the encoder. This means each position element of the decoder can attend to all position items of the input sequence.

The transformer also uses residual connection and layer normalization to improve training stability and performance. The residual connection means the original sub-layer output is recast into $SubLayer(x) + x$, which allows the model to skip over certain layers during training. The layer normalization [36] is to reduce the internal covariate shift that can occur during training.

One draw-back of solely attention algorithm is lack of the capability to make use of the position order of the token in the sequence. Therefore in a transformer model, there are position encoding added to the input embedding before feeding to the encoder, respective decoder, stacks.

2.5 Transfer learning (TL)

Transfer learning is a powerful technique in machine learning where a pre-trained model based on a large dataset, is used as a starting point for a new task. Instead of starting the training process from scratch, the pre-trained weights of the model are used as the starting point for training on a new target task or under a new domain during fine-tuning. The shared knowledge from pre-training can help the model to learn faster and more effectively, hence the task specific dataset used for fine-tuning can be comparably smaller.

2.5.1 Common TL types

Transfer learning has become increasingly popular in machine learning and has been shown to achieve state-of-the-art performance in various tasks, especially since the introduction of the transformer architecture and pre-trained large language models. Here are some common application-types of transfer learning in NLP:

- *Domain adaptation*: In this approach, a model is used to learn features from a source domain, and the knowledge gained is then transferred to a new target domain that has a different distribution and characteristics.

- *Multi-task learning*: This involves training a single model to perform multiple related tasks that can share common features. Through the shared layers, a model can utilize the common features to learn to solve several different tasks during the same training process. The T5 model is one example.

- *Cross-lingual transfer learning*: This involves transferring knowledge across different languages. For example, a model trained on English text can be used to improve the performance of a model on a related task in another language. One example is the mT5.

- *Feature extraction transfer learning*: In this approach, the pre-trained model is used as a fixed feature extractor. Only the last few layers of the model are replaced with new layers, to train for the new task. The remaining layers are are frozen and used to extract features from the new data.

2.5.2 TL: related work in NLP

Very often, transfer learning techniques refer to adapting the ready-trained model to a new domain or a new task, such as the experiments performed by Kalyanpur et al. [18].

On the other hand, cross-lingual transfer-learning is comparably more challenging. Traditionally, it requires introducing other techniques to "translate" between source and target languages, e.g. cross-lingual word embedding. In recent years, cross-lingual transfer has become a popular topic [37]. Several studies have demonstrated that direct transfer-learning from source language to the target one can, through retaining the trained weights, speed-up the training process [38]. However, most of these cross-lingual transfer learning studies target only one NLP task. For SRL, involving several sub-tasks (see Section 2.2), there is almost no published cross-lingual transfer learning studies.

In this study, two types of transfer learning techniques have been experimented for training the Swedish SRL model. The first one is cross-lingual TL, by using an existing English SRL model pre-trained on FrameNet as our starting point. This type of transfer learning has been studied widely when developing a cross-lingual system for a single down-stream NLP task. For example, for Named Entity Recognition (NER) task, Bari et al. [39] had in 2020 explored cross-lingual TL with five different target languages and a pre-trained English NER as the base model.

The other one is transfer-learning for semantic parsing, by utilizing a multilingual transformer system, such as mT5 [40]. Recently, hundreds of studies have shown it is effective ([41], [42]) for developing a Text-to-Text model with low resources when using transfer-learning with pre-trained mT5 model.

2.6 T5 and mT5

One of the chosen transfer models for developing our Swedish SRL model is based on the *Text-to-Text Transfer Transformer* (T5). The other one is a *Multilingual T5* (mT5) model. Both of them were developed by Google researchers in 2020.

2.6.1 Text-to-Text Transfer Transformer (T5)

Unlike other common models that are task-specific and require specialized architectures for each task, T5 is designed to perform a wide range of text-based language problems using a single unified model of the same architecture and weights [43]. Based on transformer encoder and the decoder architecture, the T5 model is first pre-trained on a large corpus of text data[7] by using a variant of the masked language modeling (MLM) task, where a random subset of the input tokens is masked and the model is trained to predict the missing tokens.

Once the pre-training is completed, the model can be fine-tuned on a task with treating it as sequence-to-sequence problem. Typically, the input sequence is the original input string with adding a textual prompt on the top to specify the task, and the output sequence is the target solution to that task in a text format. For example, when training on summarization task, the input can be formed as: "**summarize** : {the long input string}", where "summarize" is the prompt, and the target output is the gold summary text.

T5 comes with various model sizes: "T5-small", "T5-base", "T5-large", "T5-3b", and "T5-11b", containing 60M to 11B parameters (*more details*). Thanks for its flexibility and good generalization capability, T5 has become a promising architecture for multi-task-learning within NLP field.

2.6.2 The T5-based English SRL

To develop our Swedish SRL using the transformer technique and being able to handle all the three sub-tasks in an end-to-end manner, it is advantageous to choose an encoder-decoder transformer, such as T5 ([16], [31]). As mentioned earlier, due to the small size of SweFN dataset, cross-lingual transfer on an English SRL model can be an alternative. However there are very few English SRL covering the three subtasks in a pipeline that are published as open-source. To address this issue, Chanin has recently developed an English SRL model, *Frame Semantic Transformer*.

This English SRL model is developed through fine-tuning a pretrained T5 model, "T5-small" and "T5-base" respectively, from Huggingface ([44]), where the T5-base is the default base model. Before fine-tuning on the FrameNet 1.7 fully annotated train set, this model has been pre-trained first with PropBank data and then with FrameNet 1.7 exemplars.

[7]The C4 dataset (Colossal Clean Crawled Corpus) with 750 GB of English text is introduced together with the T5 model in paper [43]

To further increase the performance, task-specific prompt hints are added into the input sequence during fine-tuning for "*Frame Classification*" and "*Argument Extraction*" sub-tasks. More details are given in Section 3.3.

Thanks for pre-training on the related datasets and providing hints to the model during inference, the *Frame-Semantic-Transformer-base* model (220M parameters) achieves good F1 scores on trigger identification and frame classification sub-tasks, and a new SOTA performance on argument extraction. Even the "small" version (60M parameters) of the English SRL has demonstrated surprisingly good performance.

Furthermore, this English SRL model can be easily installed as an open-source Python library, and the programming code can be accessed on its Github page. It is also designed as such that all required models and datasets are downloaded directly from Huggingface model hub and NLTK [45] corpora when the model is first run, which makes it easier to be adapted for training with another model or corpora.

2.6.3 Multilingual Text-to-Text Transfer Transformer (mT5)

mT5 is a multilingual version of T5 and has also five model-size-variants. By using a similar approach as T5, i.e. MLM, mT5 is pre-trained on a large multilingual corpus of text data (mC4[8]) covering 101 languages, which provides mT5 substantial cross-lingual transfer learning capabilities. It means a mT5 model fine-tuned on NLP tasks in one language can perform the same tasks in several other languages without requiring further training on these languages [40]. This becomes one of the biggest advantages of transfer learn a mT5 model.

Unlike T5, the published mT5 model is not fine-tuned on any specific NLP task, which means the mT5 model needs supervised learning[9] before it can be used for solving a downstream task, such as summarization, NER, machine translation, etc.

The mT5 has also five variants in term of the model size, from the smallest "mT5-small" (300M trainable parameters) to the biggest "mT5-XXL" (13 billion parameter), much heavier than T5 variants.

2.7 Evaluation metrics

As illustrated in Figure 2.4, in a classification task, each prediction can be categorized as one of four when comparing with the ground truth: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Since a SRL model involves sub-tasks that are either labelling or tagging a position, the common evaluation metrics used in classification tasks, such as accuracy and F1 score, are typically chosen for evaluating the performance of a SRL model.

Accuracy measures how often the model makes correct predictions, and is calculated

[8] mC4 is a multilingual variant of the C4 dataset.

[9] That is training the model with dataset of input-output pairs, and the output is the target.

as follows:

$$Accuracy = \frac{TP + TN}{number\ of\ predictions} \tag{2.1}$$

While accuracy is a simple and intuitive evaluation metric, it may not be appropriate for all types of datasets, especially those with imbalanced classes. For example, in a dataset where one class is much more dominant in a dataset, a model that simply predicts the majority class for all examples may achieve a high accuracy score but may not be useful in practice. Therefore, when developing a SRL model, it is more common to use other evaluation metrics, such as F1 score and its underlying scores, precision and recall, to assess the performance more comprehensively ([1], [8], [18]).

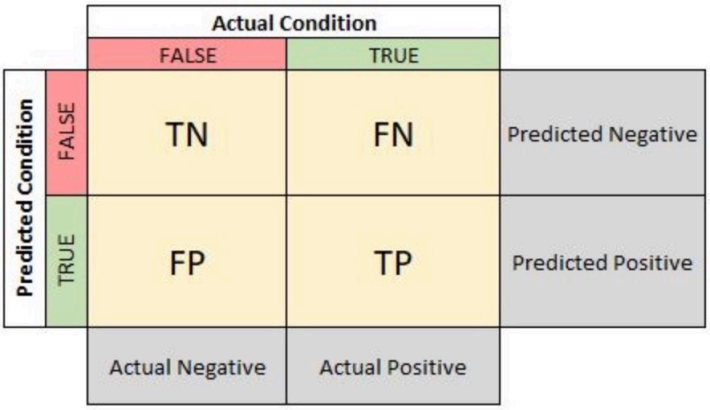

Figure 2.4: Confusion Matrix: the predicted class versus the true class in a classification task. *Source.*

Precision measures how many of the predicted positive instances are actually positive, while recall measures how many of the actual positive instances are correctly predicted as positive. F1 score represents the balance between precision and recall, thus summarizes the classification performance. It is calculated as shown in the Equation 2.2:

$$\begin{gathered} Recall = \frac{TP}{TP + FN}, \quad Precision = \frac{TP}{TP + FP} \\ F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \end{gathered} \tag{2.2}$$

The F1 score ranges from 0 to 1, where 1 represents perfect precision and recall, and 0 represents the worst possible performance. A high F1 score indicates that the model has a good balance between precision and recall, meaning that it is correctly identifying true positives while minimizing false positives and false negatives.

When using these metrics for evaluating SRL performance, it is more complicated than for a standard classification task. For example, there is no counts of TN, and

for “argument extraction” sub-task, a TP requires both positioning and labelling of the FE are correct. Detailed description on the counting of TP, FN and FP for each SRL sub-task are provided in Section 3.5.1.

3

Method

3.1 Transfer models

3.1.1 Model 1: cross-lingual TL

In term of cross-lingual transfer learning, although a multilingual SRL model would be a more suitable choice, such model is not available, plenty studies have demonstrated that a English T5 model can cross-lingual transfer to other languages surprisingly well ([46], [47], [48], [49]). This may be explained by that a English T5 model has actually exposed to a small amount of non-English text[1] during its pre-training process and the language similarity of the new target language to English [50].

As described earlier in Section 2.6.2, the T5-based *Frame Semantic Transformer* [16] has several considerable advantages, such as, covering three sub-tasks in a pipeline fashion, state-of-the-art performance, open-source and easy-of-use. Plus the cross-lingual transfer capability of a T5 model, this English SRL model has therefore become an obvious choice of the primary transfer model in this study.

As the "base" version of the English SRL has shown much more improved performance on argument extraction task than the "small" version, it is therefore preferable to choose the "base" model variant only. But to speed-up the training setup and quickly gain experience of a Swedish SRL model, we have decided to start with the "small" version, and then try with the "base" variant. Our Swedish SRL models developed from this approach are called MODEL-1-SMALL and MODEL-1-BASE respectively. Due to missing an existing Swedish SRL model to benchmark, we choose MODEL-1-SMALL as the baseline in this study.

However, we have to remember that a T5 model is not the best choice for cross-lingual transfer due to the lack of strong relationship among multiple languages established through significant exposure to these languages. Besides, Swedish special letters such as *ä, å, ö* are not in the vocabulary of T5 tokenizer by default, and these unknown chars will be converted to "<unk>". The latter may cause it harder for the model to learn the relationships among Swedish words containing special letters.

Further, in term of implementation, this English model has been trained only identify the starting position of the trigger in a sentence, which is not ideal (see Section 3.4.1).

[1] Indeed, C4 actually contains 0.22% text of other languages than English [50].

Thus, the Swedish SRL transferred from this English model, or trained by the same implementation, will keep marking only the start position of a trigger.

3.1.2 Model 2: multi-task TL

Inspired by the English SRL model, especially the capability of perform all the three sub-tasks directly in one shot, we would like to deploy a Swedish version of T5 as the second transfer model for transfer-learning on SRL tasks. In absence of such model, the multilingual T5 (mT5) has become a possible option.

This alternative choice has its advantage as well. After pre-trained on more than 100 languages that include English and Swedish, a mT5 model has established weight-parameters that can represent the dependency between English and Swedish words. And in the SweFN corpus, both the frame names and the FEs names are in English, meanwhile the input sentence and the lexicon-units are in Swedish. Therefore, the Swedish SRL model actually needs to "understand" the relationship between these two languages. A mT5 model, with already obtained attention score between the targeted English names (Frame names or FEs names) and the relevant Swedish words in the input sentence, has indeed an advantage for the multi-task TL on SRL sub-tasks.

The authour of the mT5 (see Section 2.6.3) model has shown that the largest version of mT5 ("mT5-XXL") can outperform previous popular multilingual models for most NLP tasks [40]. Therefore, "mT5-XXL" is much more preferable. However, as the mT5 is pre-trained on more than 100 languages, even its smallest version ("mT5-small") is five times bigger than the "T5-small". Due to the constraints of our computer capacity, it is hard for us to train a bigger variant of mT5. Therefore, in this study, we use only the "mT5-small", from which the further developed Swedish SRL model becomes the Model-2-small. Experimenting with the small model variants can speed up the training process and still be able to provide a fair benchmarking between these two TL approaches.

Besides, since the mT5 has not yet gone through any supervised training to learn any NLP downstream task, direct fine-tuning on a small dataset (such as SweFN data) is not optimal, and will also need a longer learning process than a model has been pretrained by multiple downstream tasks, like T5.

The lack of supervised pre-training, plus unnecessarily large size than needed for the Swedish SRL, make a mT5 model less computational cost efficient than a T5 model. This becomes the main disadvantage of deploying a mT5 to develop our Swedish SRL model.

3.2 SweFN Dataset

The Berkeley FrameNet v1.7, available from *nltk.corpus*, can be directly used for generating training examples of input sequence to target sequence. But the SweFN data file is only accessible in *XML* format. For example, the frame "Absorb_heat" has its listed LUs and FEs shown in Figure 3.1 and an annotated sentence, "*Vet*

du att vatten kokar redan vid 70°C när man är på toppen av världens högsta berg, Mount Everest?", in Figure 3.2.

```
<LexicalEntry xml:lang="swe">
  <Lemma />
  <Sense id="swefn--Absorb_heat">
    <feat att="BFNID" val="Absorb_heat" />
    ...
    <feat att="LU" val="puttra..1" />
    <feat att="LU" val="torrkoka..1" />
    <feat att="LU" val="koka..4" />
    ...
    <feat att="LU" val="sjudning..1" />
    <feat att="coreElement" val="Container" />
    <feat att="coreElement" val="Entity" />
    <feat att="coreElement" val="Heat_source" />
    ...
    <feat att="peripheralElement" val="Cause" />
    <feat att="peripheralElement" val="Circumstances" />
    <feat att="peripheralElement" val="Depictive" />
    ...
    <feat att="peripheralElement" val="Time" />
```

Figure 3.1: SweFN XML file: one example shows the features associated with frame "Absorb_heat". The attributes "LU" indicates the triggers, and "peripheralElement" are the "non-core FEs".

```
<karp:example>
  <karp:text>Vet du att</karp:text>
  <karp:e name="Entity">vatten</karp:e>
  <karp:e name="LU">kokar</karp:e>
  <karp:e name="Temperature">redan vid 70 °C</karp:e>
  <karp:e name="Circumstances">när man är på toppen av
  världens högsta berg, Mount Everest</karp:e>
  <karp:text>?</karp:text>
</karp:example>
```

Figure 3.2: SweFN XML file: one annotated sentence example under frame "Absorb_heat".

Hence, we need extracting annotated sentences and frame components from this XML file before being able to further process the data into input-target sequence pairs for the training. As usual, *ElementTree* is applied for parsing XML data.

After obtaining XML frame tree, we use function *extract_frame* (see Appendix 1) to collect frame components for each semantic frame, which is then stored as a *Frame* class containing the frame name, core FEs, non-core FEs and LUs attributes. Due to the LU-definition-constraints from SALDO (see Section 2.1), in the SweFN corpus, some sentences have marked triggers that do not appear in the list of LUs for the frame, but are listed as "suggestionForLU" instead. Therefore, We have chosen to include all suggestionForLU in the list of LUs under the frame when training the Swedish SRL in this study.

Using function *extract_example* (see Appendix 1), we further collect each example sentence as a FrameAnnotatedSentence class, which contains both the sentence

as plain-text and the frame semantic annotations as of a FrameAnnotation class. This class contains the *frame name*, the *trigger location*, and the semantic annotation of frame elements in a *FrameElementAnnotation* class that contains both the labelling of each frame-element and its start- end-index position inside the sentence. Here, the *trigger location* stores only the starting position of the trigger, not the end location, which may not be the best choice of application. More information is given in Section 3.4.1.

All above mentioned classes[2] were created by Chanin when training the English SRL on BFN[16]. Using these classes enables the adaptation of the BFN data_loaders to the Swedish data_loaders that take the SweFN dataset instead of the FrameNet 1.7.

Below example shows how the sentence, "*Vatten kokar vid lägre temperatur än 100 grader Celsius om lufttrycket är lägre än normalt .*", get its extracted annotations stored in the FrameAnnotatedSentence class:

```
FrameAnnotatedSentence(text='Vatten kokar vid lägre temperatur än
↪ 100 grader Celsius om lufttrycket är lägre än normalt .',
↪ annotations=[FrameAnnotation(frame='Absorb_heat',
↪ trigger_locs=[7],
↪ frame_elements=[FrameElementAnnotation(name='Entity',
↪ start_loc=0, end_loc=6),
↪ FrameElementAnnotation(name='Temperature', start_loc=13,
↪ end_loc=55), FrameElementAnnotation(name='Circumstances',
↪ start_loc=56, end_loc=90)])])
```

3.3 Training set-ups

3.3.1 Training procedure

When choosing the *Frame Semantic Transformer* as the primary transfer model in this study, one of the biggest advantage is both the trained model and the training procedure are entirely open-source and easy-of-use [16]. Indeed, to facilitate the continual training on this English SRL model, the training program can take another dataset than the default FrameNet 1.7, through user-defined data-loaders. Therefore, when developing our Swedish SRL model, the English SRL implementation procedure by Chanin [16] has been closely followed, along with necessary adaptations to Swedish and the SweFN dataset. This approach speeds up the set-up of training the Swedish SRL model, and makes the results more comparable with the English SRL model on BFN.

To generate the training data (input-target pairs), it needs simply call the function "tasks_from_annotated_sentences" imported from "frame_semantic_transformer" [16]. This function takes both *SwedishTrainingLoader* and *SwedishInferenceLoader*

[2]More details of these classes are available at `https://github.com/chanind/frame-semantic-transformer`

(see Appendix 1) as inputs to generate the "Input/Target" sequences of the 3 sub-tasks. The *SwedishTrainingLoader* is to load example sentences during the training and validation process. The *SwedishInferenceLoader* is to buld the task-specific hints that tell the model available frames and LUs during inference. The normalization of trigger words and LUs (see Section 3.4.2), *normalize_lexical_unit*, is therefore included in the *SwedishInferenceLoader* (see Appendix 1).

Since our development procedure are based on the one of the English SRL model that using PyTorch Lighting framework, we can simply call the existing "*train*" function (shown in Appendix 1) to train and evaluate the Swedish SRL model. More details of the task-generating methods of each SRL sub-task are described in the Section 3.4.

3.3.2 Data augmentation

When loading the training dataset from BFN corpus, there are data augmentation operations deployed by the English SRL model. The applied augmentations include: misspelling (either randomly, or keyboard-based typos), synonyms-swapping, quotation-marks-replacing, upper-/lower-casing, punctuation-deleting, and contraction-removing. These augmentation techniques are used to improve the robustness of the trained model when facing the real world data that may contain errors and typos, although they do hurt the model performance a bit in Argument Extraction task according the author.

Ideally, we would like to explore which Swedish data augmentation techniques most suitable for our training. But due to time and scope limitation, we did only included the simplest augmentations, upper-/lower-casing and punctuation-deleting, in this study.

3.3.3 Choice of hyperparameters

A major part of training is done with single GPU of GeForce GTX 1070 with 8GB memory. Due to the memory constraints, when developing the larger model, Model-1-base and Model-2-small, a smaller batch-size of 4 was applied instead of 16 used for training Model-1-small.

When fine-tuning on already fine-trained large language model, it is common to apply a rather small learning rate (LR). For Model-1, we continue use the LR of 5e-5, same as the English SRL [16] trained with. When developing Model-2, since "mT5" has not yet been fine-tuned for any specific NLP task, a comparably higher LR (0.001 with drop rate 0.1 is applied in Raffel's mT5 fine-tuning paper [40], [43]) is recommended. After some experiments, we have finally chosen a decreasing LR: 5e-4 (epoch 0-15) $\longrightarrow$ 1e-4 (epoch 16-20) $\longrightarrow$ 5e-5 (epoch 21-30), which gives a better fine-tuning results than a constant LR of 5e-4.

3.4 SRL sub-tasks

Since a T5 model requires the problem being presented as a sequence-to-sequence task, the three sub-tasks have been first reformed as the following format in the input sequence: "<task name> <task-specific hints>: <text>". This format was introduced by Chanin in the English SRL model, where he claims the extra in-context prompt <task-specific hints> can facilitate the training of the model. Below are the detailed description of how each sub-task being formatted as a sequence-to-sequence task.

3.4.1 Trigger identification

For a given example sentence, the *trigger identification* task is simply to predict the position of a trigger that have evoked a semantic frame in the sentence. Therefore, this task is rather simple and do not need any task-specific hints, thus the input sequence is structured as "TRIGGER: <text>"

It is worth to mention that, in the implementation of the English SRL model, the author has chosen to use a asterisk character, *, to mark-out only the start-position of the trigger, not the end-position. Such as in the following example, * is placed just before the trigger, "*bravera*", in the target output of the given sentence: "*En förmiddag, när hon skulle * bravera med sin konst, störtes hon av Rosa, en fundamentalist.*". This is not ideal. Because when a trigger contains multiple words, with * placing before the first trigger word, it is not clear what exactly is the identified trigger. For example, in the sentence below, one may take the identified trigger as of the single word "*lättar*", but here it is actually double-words, "*lättar ankar*".

```
Input: "TRIGGER: Ostindiefararen Götheborg lättar ankar för sin
↪  färd till Kina."
Target: "Ostindiefararen Götheborg * lättar ankar för sin färd
↪  till Kina."
```

The author did not explain exactly the reason of choosing only marking-out the start-position of a trigger. But, as the English SRL model is trained with fully annotated sentences that may contain multiple frames, this sub-task will target on identifying every trigger for each evoked frame in the given sentence. In current implementation, this means having * marked in front of each trigger in the training target. If * had been applied on either side of the trigger, the output would contain double number of * with mixed start- and end-positions, which may increase the complexity in the following sub-tasks that require separating triggers to generate one input sequence for each frame.

Not like the BFN with possible multiple frames annotated within one sentence, each sentence of SweFN contains only annotations for one single frame. Without introducing new possible triggers in the annotation of a sentence, the *trigger identification* task can not be fully trained through this implementation, and the Swedish SRL model may only ever identify one trigger for one frame in a given sentence, thus less meaningful.

But, this study aims to create an End-to-End semantic role labeller that can simply take a sentence as input and perform all the three sub-tasks, including prediction of the trigger, in a pipeline fashion. Moreover, these three sub-tasks are trained independently, thus including or removing the *Trigger identification* task will not affect the training results of the other two and keeping this sub-task can only be a bonus at no extra cost. Therefore, in this study, we have decided to keep the three tasks pipeline training implementation that used in the English SRL model.

3.4.2 Frame classification

The frame classification step is to predict the frame name that evoked by the identified trigger. Following the English SRL implementation, this task is trained on one identified trigger at a time, thus N number identified triggers of a sentence together their respective triggered frames will result to N number input-target training sequence-pairs.

With the BFN train set, this implementation allows the model to predict each annotated frame within a sentence, and every identified trigger will provide one training task sample for predicting the frame name, which results to much less training samples generated for the *trigger identification* than for other two sub-tasks. To address the imbalance between sub-tasks, in the English SRL model, Chanin had repeated training samples for *trigger identification* task to force an approximate balance of task types [16].

When training the Swedish SRL model, since 95% SweFN example sentences have only one trigger annotated, it is rather balanced among task types. Thus, no need of task-balancing technique of duplicating training samples. On the other hand, due to 5% sentences of the SweFN dataset contain multiple triggers annotated for a single frame, one sentence will create multiple training examples with each trigger and the same target frame, which may actually worsen the imbalance of frames a bit.

For *frame classification* task, the <task-specific hints> is a list of frames that can be triggered by the identified trigger in the sentence. For the English SRL model, it is created through looking up the trigger with the LUs registered under each frame in the BFN corpus. Chanin claimed that this list appeared to be a powerful hint for the presence of a frame even though the registered LUs may not be exhaustive [16] for a frame. For this lookup, both trigger words and frame LUs are normalized first in a same way that including lower-casing, stemming and lemmatization. It is worth to mention, the English SRL model has deployed four stemmers and lemmatizers from NLTK to gain a higher chance of matching, which may lead to four normalized versions [16].

When training the Swedish SRL model, we have adapted this implementation for the Swedish words and dataset. Our normalization starts also from lower-casing. In SweFN, each LU ending with "**..**" plus a number or a letter and more (see examples in the Figure 3.1), which is to distinguish a word with multiple meanings as multiple LUs. For instances, *koka..4* is for frame "Absorb_heat" and *koka..1* is under the frame "Apply_heat". Removing the sense identifier can help the trigger word

koka to match with their reformed lexical units in SweFN. Afterwords, we remove all symbols other than the Swedish letters, and then stem the trigger word and lexical units by the Swedish Snowball stemmer (explained *here*).

A LU may contain multiple words, such as "*lätta_ankar..1*" and "*lägga_ut..2*" in frame of VEHICLE_DEPARTURE_INITIAL_STAGE. Since there is only one marked trigger word, bi-grams checking has been deployed in the English SRL model for this task type to increase the chance of finding the matched frames. Bi-grams are created by adding the words on either side of the trigger word and are normalized by the same approach as for the trigger word.

Bi-grams checking has an essential role in training *frame classification* task, because the listed frames hints will increase the chance of being the predicted output of the T5 model. Therefore, this implementation has also been applied in the Swedish SRL model, with our Swedish modified normalization function.

For an example, after the trigger word "*lättar* " being identified in the sentence "*Ostindiefararen Götheborg * lättar ankar för sin färd till Kina.*", both the mono-word trigger "*lättar* " and its bi-grams "*lättar ankar*" and "*Götheborg lättar*" will be normalized and become "*lätt*", "*lätt_ank*" and "*götheborg_lätt*" respectively. These normalized potential triggers overlap with normalized LUs from four frames, which are then added into the prompt to provide hints for predict the frame name. The final generate train task sample is demonstrated as below.

```
Input: FRAME Vehicle_departure_initial_stage Difficulty
↪ Emanating Experiencer_focus : Ostindiefararen Götheborg *
↪ lättar ankar för sin färd till Kina.
Target: Vehicle_departure_initial_stage
```

3.4.3 Argument extraction

The argument extraction is to identify the locations of all FEs and label them with the corresponding semantic roles for the frame in question. When performing arguments labelling, to narrow down the number choices of possible semantic roles, we provide the model with a list of all FEs available in SweFN lexical database for the given frame.

Following the procedure described in Chanin's paper [16], we treat this supporting list as part of the prompt input, so that this task is of form "<task> <targeted frame> | <all available core and non-core FEs for the frame in question> : <target sentence with the trigger marked>". The output of this sub-task is of form <FE i>="<argument i>" following by "|" to separate with next semantic role extraction. Following the same example as above, this argument extraction task is generated as following:

```
Input: ARGS Bragging | Expressor Medium Message Speaker Topic
↪ Addressee Degree Depictive Internal_cause Manner Means
↪ Occasion Particular_iteration Place Reason Role Time : En
↪ förmiddag, när hon skulle * bravera med sin konst, störde
↪ hon av Rosa, en fundamentalist.
Target: Time = En förmiddag, när | Speaker = hon | Topic = med
↪ sin konst
```

3.5 Evaluations

Since one annotated sentence parallel provides 3 generated tasks for respective 3 SRL sub-tasks, hence each of the three SRL sub-tasks are trained independently. This gives an advantage of being able to evaluate the model on each SRL sub-task separately. Within the train data loader, SwedishTrainingLoader, we randomly shuffle the generated training tasks and use 80/10/10 ratio of the train/validation/test split. This split is recommended by Chanin [16] and is commonly used by previous state-of-the-art English frame semantic parser ([21], [18]). The validation set is deployed during the training to monitor the development progress and discover eventual under-training or over-training issue. We use the held-out test set only for final evaluations and analyses of our model performance.

Even though we cannot benchmark directly with the state-of-the-art English SRL models that trained on the English FrameNet, BFN, we still show its reported F1 scores as a good reference to identify the gap for future work.

3.5.1 Quantitative analyses

Same as the English SRL model, we calculate the F1 score (see Section 2.7) on each sub-task as following [16]:

- For "trigger identification" sub-task: it counts a TP when a trigger location is market correctly, and each incorrect location counts a FP. A missing identified location counts a FN.
- For "frame classification" sub-task: with giving the gold trigger as part of the input, a correct classified frame counts a TP, and an incorrect classification counts both a FN and a FP.
- For "argument extraction" sub-task: with providing the gold trigger and frame name in the input, each correct identified and labeled FE counts a TP. Each incorrect marked and labelled FE counts a FP. A missing identified FE regards as a FN. Each partial error, either in identifying the FE location or in labelling the FE, is regarded as a FN and a FP.

For deeper analysis, we calculate F1 score per semantic frame against the test set, to identify on which frames are most easy or hard for our model to predict.

3.5.2 Qualitative analyses

In addition to F1 score quantitative metrics, for gaining concrete knowledge of the strong and weak points of the trained models, we conduct manual assessment with a few examples selected from the test set.

One sentence end-to-end prediction
Here, the model draws inference for the 3 sub-tasks in a sequence, thus giving a non-annotated sentence as the input, the model will output its prediction on all sub-tasks in a order. Thus, this type of evaluation involves chain reaction. If a trigger is wrongly identified, the "frame classification" will be based on this wrong trigger. If the "frame" is incorrectly "classified", then the "argument extraction" will be done for the incorrect frame as well.

Since a different allocated trigger than the "gold trigger" may also be correct, systematical analysis of the final output requires linguistic expertise, and therefore is not in our focus. Finally, we have chosen one short sentence for illustration.

Easiest and hardest frames to predict
For this type of evaluation, the focused frames are selected based on their performance scoring, highest and lowest possibility of wrong labelling. Through looking into their annotated samples, we try to understand better the underlying reasons of the training results.

4

Results

4.1 Quantitative results

After fine-tuning 61K steps (43 epochs) for Model-1-small, 79K steps (14 epochs) for Model-1-base, and 152K steps (27 epochs) for Model-2-small, we observed the best performance of the respective model. These best results per model type and sub-task type are summarized in Table 4.1.

F1-score (val/test)	Trigger ID	Frame CL	Argument EX
Model-1-small	0.50/0.50	0.60/0.60	0.52/0.53
Model-1-base	0.52/0.47	0.62/0.63	**0.57/0.58**
Model-2-small	**0.56/0.52**	**0.64/0.67**	0.54/0.55
English-SRL-small [16]	0.74/0.70	0.83/0.81	0.68/0.70
English-SRL-base [16]	0.78/0.71	0.89/0.87	0.74/0.72

Table 4.1: Model comparison through F1 scores on SweFN validation (val) set and test set respectively, reported per task type: trigger identification (Trigger ID), Frame classification (Frame CL), and Argument extraction (Argument EX).
The last 2 rows are the reported F1 scores on BFN dataset of the English SRL models that referred in this study.

Although, in term of F1 scores, there are still significant gaps comparing to the SOTA English SRL models ("Frame_Semantic_Transformer" [16]), our Swedish SRL models have reached acceptable performance when taking into account of the more challenging dataset (SweFN) we have. Especially, on the most important sub-task "Argument extraction", the bigger model developed from the English SRL (Model-1-base) has out-performed other Swedish SRL models and remain less gap to the aimed performance of the English SRL model on the English FrameNet data.

4.1.1 Model 1: fine-train the English SRL

Figure 4.1 and Figure 4.2 show the development when fine-tuning the SOTA English SRL models. As we can see in the upper graph of each figure, the validation loss starts to increase after a while, even though the F1 scores are continually improving

on validation set. Note, it is not overfitting as long as the accuracy is still increasing on unseen data.[1]

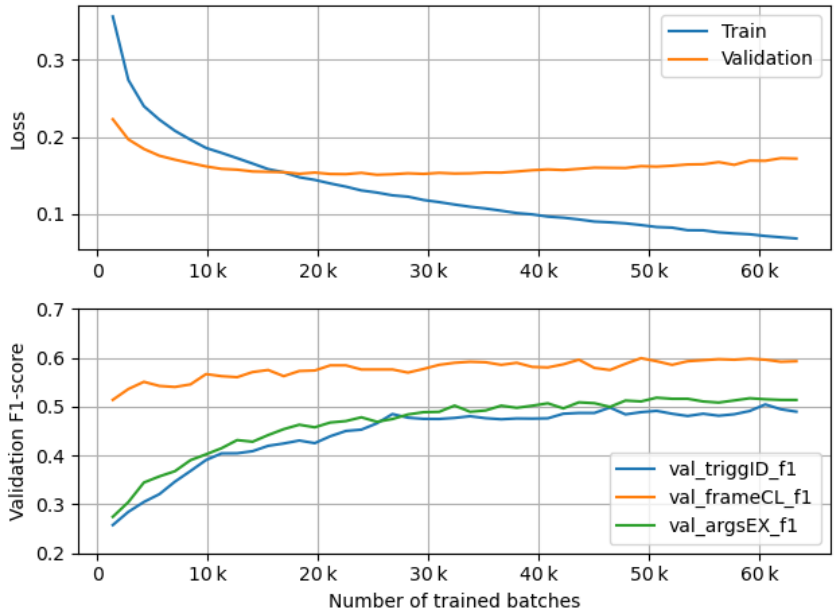

Figure 4.1: Model-1-small: loss development (UPPER) and validation F1-score for each sub-task (LOWER) during 45 epochs with batch-size of 16.

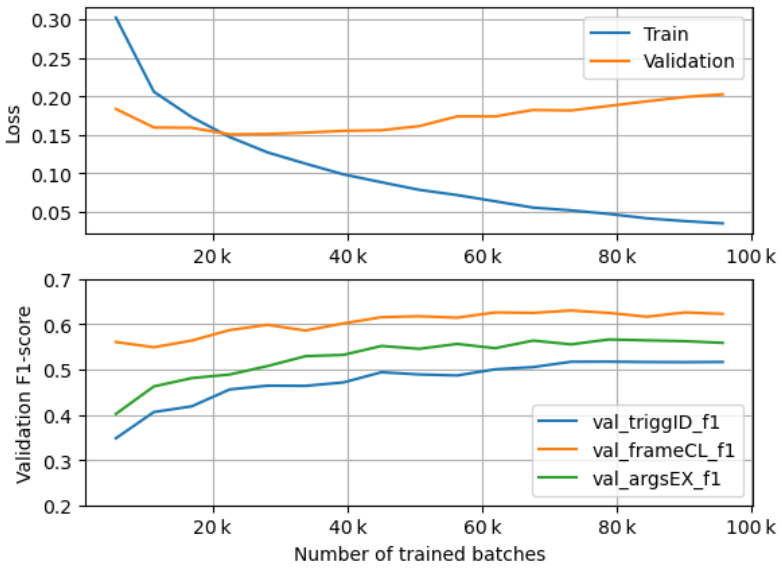

Figure 4.2: Model-1-base: loss development (UPPER) and validation F1-score for each sub-task (LOWER) during 17 epochs with batch-size of 4.

[1]T5 (as well as mT5) uses the standard Cross-Entropy loss function to compare each token of the model predicated sequence against the target sequence. Cross-Entropy loss is commonly used by any language model due to its mathematical property, and is explained *here*. While the confusion metrics used for evaluation of the model performance, are entirely different calculations than Cross-Entropy loss, and therefore do not always follow the same developing trend.

4.1.2 Model 2: fine-tune a mT5

As we can see from the Figure 4.3, because the base model mT5 has not been fine-train only any NLP downstream task before, we expect that the F1-scores start from a much lower level than the previous two models. It is the case on all sub-tasks except for "Frame classification", the performance on which has already reached a similar level as the Model-1-small after just one epoch..

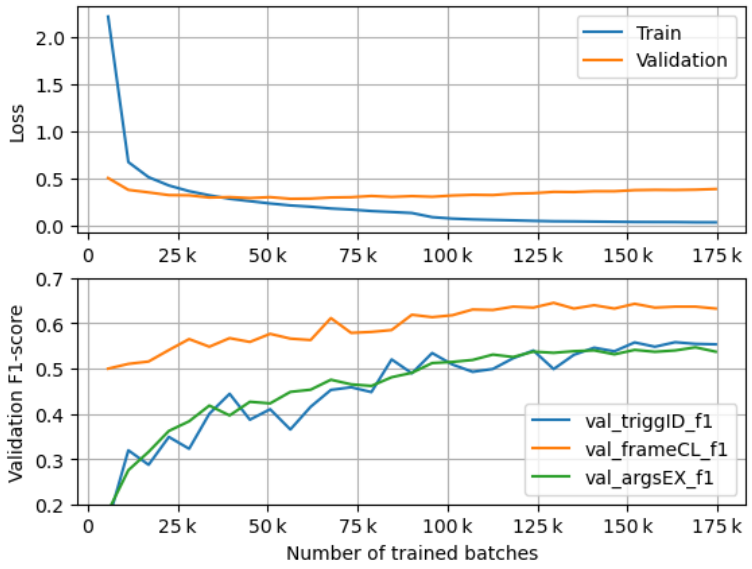

Figure 4.3: Model-2-small: loss development (UPPER) and validation F1-score for each sub-task (LOWER) during 31 epochs with batch-size of 4.

4.2 One sentence end-to-end prediction

For the example sentence: "*Axel fick parkerade på Odengatan , detta gjorde honom pepp .*", the "gold" frame semantic annotations from SweFN are as following:

```
<Sense id="swefn--Placing">
    ...
  <karp:e name="Agent">Axel</karp:e>
  <karp:e name="LU">
    <karp:e name="Manner">fick</karp:e>
    <karp:g n="1" />
    <karp:e name="LU">parkerade</karp:e>
  </karp:e>
  <karp:e name="Place">på Odengatan</karp:e>
  <karp:text>, detta gjorde honom pepp .</karp:text>
```

With using the three fine-trained Swedish SRL models, we obtain the semantic role labellings as below:

- Model-1-small

  - TRIGGER word: **parkerade**
  - FRAME: **Placing**
  - FEs: **Theme**: Axel; **Goal**: p Odengatan
- Model-1-base
  - TRIGGER word: **parkerade**
  - FRAME: **Placing**
  - FEs: **Agent**: Axel; **Goal**: p Odengatan
- Model-2-small
  - TRIGGER word: **gjorde**
  - FRAME: **Perception_active**
  - FEs: **Perceiver_agentive**: detta; **Perceiver_agentive**: honom pepp

It seems that the Model-1-base has achieved the closest results as the "golden truth", compare to which the only errors are labelling "*på Odengatan*" as "Goal" instead of "Place", and missing the FE "*fick*" as "Manner".

Regarding the miss-labelled FE "Goal", we can see some root causes from the training samples. The frame PLACING has 42 annotated sentences, 21 of them have annotation of the FE "Goal" and only 7 examples include FE "Place". Besides, when a phrase formatted like "**på** + <some object>", it is more often to have the phrase marked as "Goal" and "Place" (see Appendix 2). This may cause difficulty for the model to distinguish the semantic role between "Goal" and "Place" when a phrase starting with "**på**".

Regarding missing the FE "Manner", it may due to all of the annotated examples containing the FE "Manner" have it within a compound annotation "Manner+LU". As discussed early (in Section 5.1.1.4), it is hard for our models to learn compound annotation.

As expected, another potential trigger than the "gold" one can be marked out. In this example, "*gjorde*" is identified by the Model-2-small, hence further labelling is different in term of the evoked Frame and its FEs. However, the frame seems have been classified wrongly in this case, a proper one should be CAUSATION. This maybe because this trigger word can trigger more than 13 frames and the CAUSATION frame has only "seen" this trigger word 1-2 times out of more than 30 training samples.

4.3 Performance per frame

Out of the three SRL sub-tasks, Frame Classification and Argument Extraction tasks should be the mostly relevant for analysis per frame. As Argument Extraction is a more complex problem and harder to visualize in one graph, we choose to focus on Frame Classification in this analysis.

Intuitively, the more training samples for a frame, the better will a larger language model predict on this frame. Therefore a scatter plot (Figure 4.4) is chosen to see if there is any relationship between them. To our surprise, theses plots show that more training samples do not have a straight forward positive impact. Indeed, frames trained with almost zero samples can also have 100% prediction accuracy. But for frame with more than 20 training sentences, the major can have prediction accuracy above 50%. When comparing the prediction accuracy between Model-1-small and Model-2-small, we can see that the mT5 transferred learned model has improved performance on frames with all size of trained samples. In Section 5.3 and 5.4, we will look into these frames that have improved performance by Model-2-small, as well the most easy or hard to train frames.

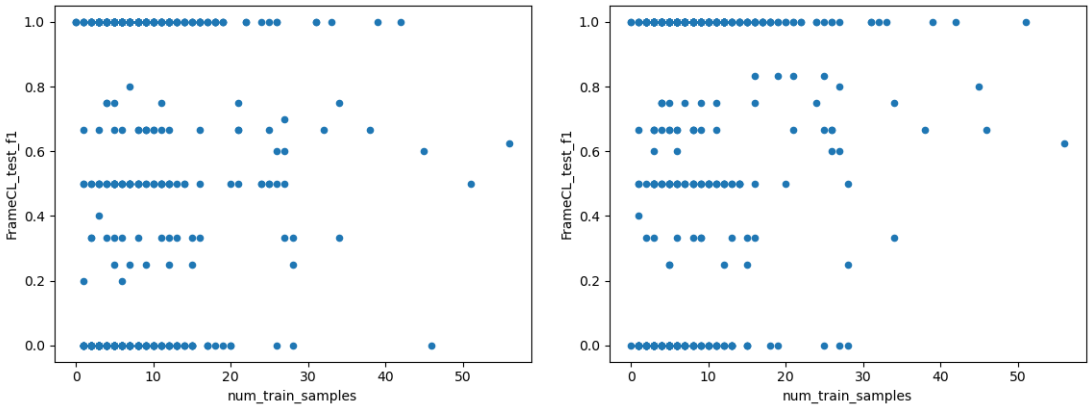

Figure 4.4: On Frame Classification task: the relationship between prediction performance (F1 test score on *Y-axis*) and number trained samples on *X-axis*. To LEFT: predicted by Model-1-small; To RIGHT: predicted by Model-2-small

To select the most extremely cases, we choose following defination:

- "Easy frames": frames with less than 3 training samples but obtaining 100% Frame Classification accuracy on test set.
- "Difficult frames": frames with more than 20 training samples but obtaining less than 60% Frame Classification accuracy on test set.

4.3.1 Easy frames

Different models may have different performance on each frame, but based on our definition there are thirteen common easy frames for both Model-1-small and Model-2-small. These frames share some common features that make them easier to get a corrected prediction, such as:

- Having very "unique" lexical units that are not listed under any other frames, such as the frames: Limitation, Judicial_body, Being_questionable, Cause_emotion, Conferring_benefit, Dynamism, Gizmo, and Typicality.

- The identified trigger having only "seen" together with the target frame. For example, EVENT_INSTANCE have all annotated sentences triggered by word "gång" or its bigrams, and this trigger word has only been trained under this frame even though it is listed under multiple frames. It is the same for other frames, like: PROCESS_RESUME, SUBJECTIVE_TEMPERATURE, SURRENDERING, and SURRENDERING_POSSESSION.

Besides, the Model-2-small has more "easy frames" than the Model-1-small, very often it is when the frame hint being empty in the input prompt, such no matching frames have been found. As the example below for the frame PRISON:

```
Input:      FRAME  : Utpekad som pådrivande bakom demonstrationen
↪  dömdes hon till nio år i det ökända tibetanska * fängelset
↪  Drapchi.
Output:     Prison
```

Although trigger word "fängelse" is one of the listed LUs, but due to the limitation of the Swedish Snowball stemmer, "fängelset" has not been transferred the same stem as "fängelse", thus no suggested frames in the hint. Due to lack of the hint, the Model-1-small failed to identify the frame, but Model-2-small can still output the correct frame PRISON. It may thanks for the learnt relationship between this trigger and the target frame in mT5 based model.

Surprisingly, there is also some frames seem to be easy for Model-1 but not for Model-2, such as the frames: COMMUTATIVE_PROCESS, LOSING_IT, ROBBERY, SERVING_IN_CAPACITY. Most often, it will happen when the marked trigger has not been seen together with the target frame in the train set, but only in the LU lists of this frame, hence in the hint list. Indeed, the English SRL based model seems more reply on the hints in the input prompt, but not the mT5 transferred model, which seems to be more capable to predict with its established dependency between the English frame name and the Swedish trigger words, just like in the following example:

```
Input:      FRAME Breaking_apart Losing_it : Igår höll jag på att *
↪  explodera.
Output:     Breaking_apart
```

Here, the trigger word "explodera" has never been seen with frame LOSING_IT in the train set, but may have a pre-trained dependency with phrase "breaking apart", which then becomes the prediction output.

However, occasionally, the Model-2-small can also make up a fake frame name through its pre-established connections among Swedish and English words. For instance, in the following case, the predicted frame EVENTIVE_PROCESS has never been in the SweFN corpus.

```
Input:      FFRAME Commutative_process : Jag var just inne på posten
↪  och * gångrade och delade i huvet.
```

```
Output:     Eventive_process
```

4.3.2 Difficult frames

Based on our definition, the frames that most challenging to predict are: Simultaneity, Removing, Filling, Cause_motion, Part_ordered_segments, and Proliferating_in_number. The difficulty can usually explained be a combination of following common reasons.

- Reason 1: Trigger is a compound word but the listed corresponding LU is of multiple words, hence no matched frames could be given in the prompt hint. For example, "vältra_bort" is a listed LU in the frame Cause_motion but its compound format "bortvältrade" are used in the test sentences.
- Reason 2: Due to limitation of Snowball stemmer, the normalized trigger and LUs do not match each other, hence empty hint in the prompt, such as the trigger "åket" and the listed LU "åk..2" for frame Part_ordered_segments.
- Reason 3: The trigger is less common and has never or very seldom been trained together with the target frame, thus no strong dependency can be established between the frame name and the trigger words.
- Reason 4: Too many matched LUs for a trigger, thus many suggested frames in the hint, but this trigger has most often trained for other frames than the target one, thus lower probability of predicting the target frame.

At the same time, as we can see in the Figure 4.4, the Model-2-small has much less difficult frames than the Model-1-small. This improved prediction usually happen when the hint list is empty in the input prompt. Indeed, mT5 based model does not only rely on the hints that much as the English SRL based model, but rather rely on the trained relationship between the trigger and the target frame, which gives it the advantage to predict the correct frame name.

5

Discussion

5.1 Matters that enhance the performance

When comparing the training progress among the three models as shown in the Figure 5.1, we can see cross-lingual transfer learning (used for Model-1) can indeed speed-up the fine-tuning process, especially on the "Argument extraction" task. Without pre-fine-tuning for semantic role labelling in English or any other NLP downstream tasks, the mT5 needs much longer training to reach a similar level of performance as than the models transfer learned from the English SRL model.

"Frame classification", as a comparably simpler task, has already an accuracy above 50/50 before any fine-tuning on the Swedish data when using the English SRL T5 networks as the transfer models, but improves slowly with increased training steps. This indicates that a longer training time may not be the main factor of a higher accuracy in this case. This is further discussed in Section 5.3.

"Trigger identification" is excluded from the deeper analysis in this section, because the referred "gold" triggers do not cover a full list of correct answers, thus the accuracy judgement on this task type becomes ambiguous.

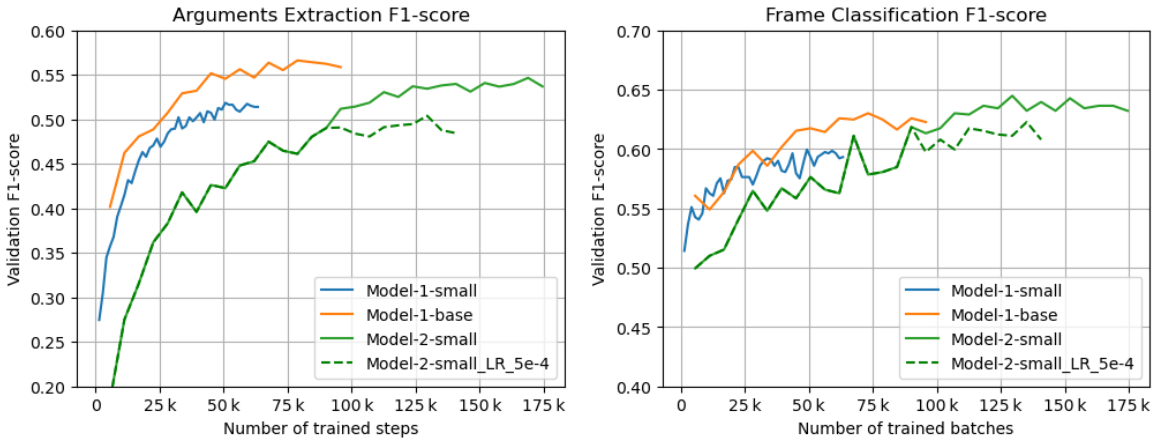

Figure 5.1: Model performance comparison: F1 score development per trained steps, in second and third sub-tasks. Note: for "Model-2-small" there are two training versions, one with constant learning rate of 5e-4, the other is the chosen version with learning rate decay (see Section 3.3.3).

This graphs approves that even the smaller variants of the T5-kind model (such as the English SRL models used in this study), also has surprisingly good cross-lingual transfer learning capability.

It shows as well that multi-task transfer learning is less efficient than multi-lingual transfer learning approach when developing a Swedish SRL model with SweFN. This may because SRL involves several sub-tasks, and the available SweFN data for training is comparably small. Multi-lingual transfer learning has more advantages in developing a Swedish SRL model may be due to a Swedish SRL requires natural language understanding of both the Swedish input texts and the English labels.

Bigger model size better model performance approves to be true in this study when developing the Swedish SRL model. Both Model-1-base and Model-2-small have achieved better F1-scores on all sub-tasks. One reason is they have 4 - 5 times more trainable parameters than the Model-1-small, thus having higher capacity to store complex relationship between words of natural language, which is very important for more complex task, such as the "Arguments Extraction".

Further, the Figure 5.1 shows that learning rate decay can improve the training of the Swedish SRL model when using mT5 as base model. It is because after warming-up phase, a larger learning rate will make the model easier to over-fit on the noisy (a special case) in the SweFN data. Since the SweFN data is small and extremely unbalanced, fine-tuning a large network (such as the mT5) on this small dataset is very sensible for over-fitting.

As previously mentioned, involving a hint (the support-list) in the prompt can indeed speed-up and stabilize the training of a T5 kind of system. This is an efficient technique, especially when this support-list is short with only a few possible choices and the task is simple like "Frame classification". Because this hint list includes the target text sequence, which highlights the correlations between certain words in input sequence and the aimed words in the target sequence, thus be able to simplify the learning process of the model. With applying this technique, model size and multilingual transfer learning type becomes less important.

5.2 Data issues that obstruct the training

Single annotated Frame/LU per sentence
As pointed out earlier, the SweFN at current stage contains only example sentences with single LU and frame annotated per sentence. For instance, sentence "*Nästa år får svenskar betala tio dollar för att få resa in i USA .*" has at least 2 triggers, "*betala*" evoking frame "COMMERCE_PAY" and "*dollar*" activating frame "MONEY". Because SweFN has only marked "*dollar*" as the "true" LU in this sentence, if the Swedish SRL model predicts "*betala*" as the single trigger, it would have been treated as an incorrect location (counts a FP) and a missing identified location (counts a FN), which will lead to an incorrectly lower F1 score calculated for the "trigger identification" sub-task. Most important, without fully annotated sentences, the Swedish SRL developed from SweFN is not able to identify multiple triggers.

Unbalanced examples
Very often a frame includes a big number of LUs, but only a few of them have been covered by at least one example sentence. It is already an issue for training these LUs that have little training examples. It will be even worse if the annotated sentences are highly concentrated on very few triggers. For example, frame "ADDICTION" contains 32 listed LUs and 7 suggested-LUs. But out of 13 annotated sentences, 9 sentences have trigger '*beroende*" and 3 examples have '*missbruk*" marked as LU.

Unbalanced examples make the model easier to over-learn a few "demonstrated" triggers, thus harder to predict the potential triggers or FEs outside the example sentences, and therefore poor generalization of the trained SRL model. This data issue can be tackled through transfer-learning, such as fine-tuning a multi-lingual model that have been already trained on the BFN data with much richer annotations. These established connections learned from training on the BFN dataset can be re-used when training on the Swedish text.

Missing arguments
The support-list used in the prompt is not necessarily exhaustive. There are sometimes potential triggers or FEs outside the listed LUs or Frame Elements. For example, frame "REPAYMENT" has 4 listed FEs. One of its five annotated sentences has marked a FE "*Time*", which is an argument not listed for that frame.

It can be problematic, especially when only a small amount of annotated sentences available for this frame. Because missing from the "supporting list" of the input sequence, the model will have much less chance to learn the possible relations with limited dataset. But this issue may be addressed through in-language multi-task learning, that is training a Swedish SRL from a mT5 with mixed data from both BFN and SweFN.

Triggers with no matched LUs
In general, a trigger from an example sentence is always included in the listed LUs for the triggered frame in the SweFN, even though many listed LUs have no examples coverred by the annotated sentences. But sometimes, when the trigger have a compounded format, there can be a mismatching between the trigger and the registered LU. Such as the earlier mentioned "bortvältrade" vs. "vältra_bort", and "inskicka" vs. "skicka_in".

This should not be an issue if both formats have been seen during the training, and both formats have been included in the listed LUs. Otherwise, base on current implementation, the model will have little chance to learn the correlation between the compounded and uncompounded formats, thus bad frame classification performance.

Compound annotation
There are also compound annotations in the SweFN, such as in the following manner for frame "PLACING":

```
<feat att="compound" val="Duration+LU" />
<feat att="compound" val="Theme+LU" />
<feat att="compound" val="Time+LU" />
```

```
    <feat att="compound" val="Manner+LU" />
```

Due to the current approach of extracting annotated sentences (see function *extract_example* in Appendix 1) from the SweFN, we cannot collect annotations in a nested manner. Thus no compound annotations in the generated training tasks. Hence, the model cannot learn to output a nested annotation neither. Actually, this issue is hard to tackle when transfer learning on the English SRL model, because the BFN does not have this type of annotation thus the English SRL model has never been trained in this way.

5.3 Changes that may improve the training

As previously shown in Figure 4.4, frames have been trained more often, i.e. with more number of training samples, do not necessarily gain a better performance. Similarly, as we have seen in Figure 5.1, a longer training time can neither easily improve the prediction accuracy of the "Frame classification" task.

One of the reasons we have found during the analyses is lack of diversity in the training samples. Indeed, within the limited number of annotated sentences, very often it is the same trigger being trained for a target frame. If new training samples can be introduced though replacing the trigger with its synonym or its compounded format or vice versa, the model will have the possibility to see more varied training samples.

Furthermore, as previously pointed out, only applying the single Swedish Snowball stemmer cannot always lemmatize the words to its stem or base form well, especially for nouns with its indefinite and definite forms, or verbs with different tenses. Therefore, with utilizing multiple stemmer and lemmatizers, like the English SRL model, we can increase the chance of finding potential frames for a certain trigger, thus easier for the model to predict correctly.

6

Conclusion

6.1 Summary of the findings

Even though our Swedish SRL models have not yet reach the same performance on SweFN data as the English SRL model on the BFN data, given the limitation of the SweFN dataset, our experiments have shown promising future of developing an automatic Swedish SRL through transfer learning techniques and transformer architecture.

Though the comparison of two transfer models, multilingual transfer learning has shown more advantages in developing a SRL model, as pre-training of SRL tasks on English can speed up the training of performing semantic role labelling on Swedish, and the pre-trained relationships between English (frame names and Frame Elements) and Swedish (trigger words) can encourage the model to output a correct prediction.

As expected, for more complex sub-task, such as the "Arguments Extraction", a bigger model has shown the most advantage. And when deploy mT5 as the transfer model, since it having not been fine-trained on any specific downstream tasks, a bigger learning rate with learning rate decay will help to speed-up the training and reduce the over-fitting risk.

It shows adding the hint (support-list) in the prompt can indeed speed-up and stabilize the training, especially for the "Frame classification" task, and when the list is short. To fully utilize the listed hints, multiple Swedish stemmers and lemmatizers can be deployed in the normalization procedure to improve the matching chance between the triggers and the listed lexical units. In case of empty hint, the mT5 based model has advantages in learning a direct relationship between the marked trigger and the frame in target.

Lack of diversity and small size of the SweFN dataset, the trained models have a risk of overfitting on the limited training samples and thus poor generalization power, meaning bad performance on triggers or frame elements outside the train set. Together with improved lemmatization, applying suitable data augmentations (such as synonym, compounding/decomposing, verb-senses changing, etc) can tackle this issue and generalize to the triggers unseen in the training data.

6.2 Future work

Besides for applying more lemmatizers and useful data augmentation techniques, to further improve the prediction performance with the limited data of the SweFN corpus, one can use a multilingual-transformer-system-based pre-trained English SRL as the transfer model to develop the Swedish SRL. With combining the advantages of fully annotated rich dataset from the English FrameNet (BFN) and the advantages of cross-lingual dependency from a multilingual transformer system, one will not only be able to improve the prediction accuracy, but also be able to identify multiple triggers in a sentence.

Another idea is to change the training order of the three sub-tasks. With a sentence as a input, let the model to predict a few frames with highest probabilities, for example up to 3 frame. And then base on each predicted frame, train the model to allocate the trigger and Frame Elements of this frame inside the sentence. This approach can force the model to learn a relationship between the frame name and the whole sentence, not only the marked out trigger as in current implementation, thus force the model to "understand" better the context of the sentence. But on the other hand, this approach may require much longer training due to the huge number of possible frames in the SweFN corpus.

Bibliography

[1] D. Larionov, A. Shelmanov, E. Chistova, and I. Smirnov, "Semantic role labeling with pretrained language models for known and unknown predicates," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019, pp. 619–628.

[2] A. Wingfield and E. A. Stine-Morrow, "Language and speech.," 2000.

[3] E. Bastianelli, G. Castellucci, D. Croce, and R. Basili, "Textual inference and meaning representation in human robot interaction," in *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, 2013, pp. 65–69.

[4] G. Osipov, I. Smirnov, and I. Tikhomirov, "Relational-situational method for text search and analysis and its applications," *Scientific and Technical Information Processing*, vol. 37, no. 6, pp. 432–437, 2010.

[5] D. Shen and M. Lapata, "Using semantic roles to improve question answering," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 12–21.

[6] C. Shi, S. Liu, S. Ren, *et al.*, "Knowledge-based semantic embedding for machine translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 2245–2254.

[7] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[8] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and whats next," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 473–483.

[9] D. Marcheggiani, A. Frolov, and I. Titov, "A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling," *arXiv preprint arXiv:1701.02593*, 2017.

[10] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1127–1137.

[11] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[12] R. Johansson, K. F. Heppin, and D. Kokkinakis, "Semantic role labeling with the swedish framenet," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 2012, pp. 3697–3700.

[13] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The berkeley framenet project," in *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

[14] P. R. Kingsbury and M. Palmer, "From treebank to propbank.," in *LREC*, 2002, pp. 1989–1993.

[15] D. Dannélls, L. Borin, and K. F. Heppin, *The Swedish FrameNet++: Harmonization, integration, method development and practical language technology applications*. John Benjamins Publishing Company, 2021, vol. 14.

[16] D. Chanin, "Open-source frame semantic parsing," *arXiv preprint arXiv:2303.12788*, 2023. [Online]. Available: `https://pypi.org/project/frame-semantic-transformer/`.

[17] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Computational linguistics*, vol. 28, no. 3, pp. 245–288, 2002.

[18] A. Kalyanpur, O. Biran, T. Breloff, *et al.*, "Open-domain frame semantic parsing using transformers," *arXiv preprint arXiv:2010.10998*, 2020.

[19] C. Zheng, X. Chen, R. Xu, and B. Chang, "A double-graph based framework for frame semantic parsing," *arXiv preprint arXiv:2206.09158*, 2022.

[20] H. Peng, S. Thomson, S. Swayamdipta, and N. A. Smith, "Learning joint semantic parsers from disjoint data," *arXiv preprint arXiv:1804.05990*, 2018.

[21] S. Swayamdipta, S. Thomson, C. Dyer, and N. A. Smith, "Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold," *arXiv preprint arXiv:1706.09528*, 2017.

[22] N. Xue and M. Palmer, "Calibrating features for semantic role labeling.," in *EMNLP*, 2004, pp. 88–94.

[23] S. Pradhan, K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky, "Semantic role chunking combining complementary syntactic views," in *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, 2005, pp. 217–220.

[24] V. Punyakanok, D. Roth, and W.-t. Yih, "The necessity of syntactic parsing for semantic role labeling," in *IJCAI*, vol. 5, 2005, pp. 1117–1123.

[25] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[26] G. G. ahin and M. Steedman, "Character-level models versus morphology in semantic role labeling," *arXiv preprint arXiv:1805.11937*, 2018.

[27] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[28] E. Strubell, P. Verga, D. Andor, D. Weiss, and A. McCallum, "Linguistically-informed self-attention for semantic role labeling," *arXiv preprint arXiv:1804.08199*, 2018.

[29] A. M. Nair and K. Bindu, "Semantic role labelling using transfer learning model," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1767, 2021, p. 012 024.

[30] R. Bakker, R. A. van Drie, M. de Boer, R. van Doesburg, and T. van Engers, "Semantic role labelling for dutch law texts," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 2022, pp. 448–457.

[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[32] S. Oliveira, D. Loureiro, and A. Jorge, "Improving portuguese semantic role labeling with transformers and transfer learning," *arXiv preprint arXiv:2101.01213*, 2021.

[33] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

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

[35] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[36] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[37] S. Ruder, I. Vuli, and A. Søgaard, "A survey of cross-lingual word embedding models," *Journal of Artificial Intelligence Research*, vol. 65, pp. 569–631, 2019.

[38] J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer learning for speech recognition on a budget," 2017. DOI: 10.18653/v1/w17-2620.

[39] M. S. Bari, S. Joty, and P. Jwalapuram, "Zero-resource cross-lingual named entity recognition," 2020. DOI: 10.1609/aaai.v34i05.6237.

[40] L. Xue, N. Constant, A. Roberts, *et al.*, "Mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[41] A. Chrabrowa,. Dragan, K. Grzegorczyk, *et al.*, "Evaluation of transfer learning for polish with a text-to-text model," *arXiv preprint arXiv:2205.08808*, 2022.

[42] A. Elmadany, M. Abdul-Mageed, *et al.*, "Arat5: Text-to-text transformers for arabic language generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 628–647.

[43] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[44] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[45] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[46] K. Tran, "From english to foreign languages: Transferring pre-trained language models," *arXiv preprint arXiv:2002.07306*, 2020.

[47] Z. Chi, L. Dong, F. Wei, X.-L. Mao, and H. Huang, "Can monolingual pre-trained models help cross-lingual classification?" *arXiv preprint arXiv:1911.03913*, 2019.

[48] Z. Li, K. Parnow, H. Zhao, *et al.*, "Cross-lingual transferring of pre-trained contextualized language models," *arXiv preprint arXiv:2107.12627*, 2021.

[49] E. Gogoulou, A. Ekgren, T. Isbister, and M. Sahlgren, "Cross-lingual transfer of monolingual models," *arXiv preprint arXiv:2109.07348*, 2021.

[50] T. Blevins and L. Zettlemoyer, "Language contamination helps explains the cross-lingual capabilities of english pretrained models," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 3563–3574.

A

Appendix

A.1 Appendix 1: Main Python Codes

The function for extract frames and their components from SweFN XML dataset

```
def extract_frame(xml_frame) -> Frame:
    """
    Extract a Frame element from the Swedish XML for a frame
    """
    name = xml_frame.attrib["id"].replace('swefn--', '')
    core_elms = [
        ft.attrib["val"] for ft in
        ↪  xml_frame.findall(".//feat[@att='coreElement']")
    ]
    non_core_elms = [
        ft.attrib["val"]
        for ft in
        ↪  xml_frame.findall(".//feat[@att='peripheralElement']")
    ]
    lus = [ft.attrib["val"] for ft in
    ↪  xml_frame.findall(".//feat[@att='LU']")]
    # some examples have triggers outside the listed 'LU', but
    ↪  they are usually registerred as 'suggestionForLU'
    lus += [ft.attrib["val"] for ft in
    ↪  xml_frame.findall(".//feat[@att='suggestionForLU']")]

    return Frame(
        name=name,
        core_elements=core_elms,
        non_core_elements=non_core_elms,
        lexical_units=lus,
    )
```

The function for extract example sentences and their annotations from SweFN XML dataset

```
def extract_example(example_xml, frame_name) ->
↪  FrameAnnotatedSentence:
    """
    Extract an annotated training sentence from a Swedish FrameNet
↪   Example XML
    """
    nodes_to_extract = [n for n in example_xml]
    text = ""
    trigger_locs = []
    frame_elements = []
    while len(nodes_to_extract) > 0:
        node = nodes_to_extract.pop(0)
        # Sometimes there's nodes in nodes, the compound
        ↪  annotations.
        # In this case, push the children of this node to the
        ↪  front of the queue and keep going
        if len(node) > 0:
            nodes_to_extract = [n for n in node] + nodes_to_extract
        else:
            cur_index = len(text)
            if not node.text:
                continue
            node_text = re.sub(r"\s+", ' ', node.text)
            if node.attrib.get("name") == "LU":
                trigger_locs.append(cur_index)
            elif "name" in node.attrib:
                frame_elements.append(
                    FrameElementAnnotation(
                        name=node.attrib["name"],
                        start_loc=cur_index,
                        end_loc=cur_index + len(node_text),
                    )
                )
            text += node_text + " "
    text = text.strip()
    return FrameAnnotatedSentence(
        text=text,
        annotations=[
            FrameAnnotation(
                frame=frame_name,
                trigger_locs=trigger_locs,
                frame_elements=frame_elements,
            )
        ],
    )
```

The data_loader classes. As shown below, both of the two data loaders are a child class inheriting from *TrainingLoader* class and *InferenceLoader* class respectively.

```
from frame_semantic_transformer.data.loaders.loader import
↪  TrainingLoader, InferenceLoader
from frame_semantic_transformer.data.frame_types import Frame,
↪  FrameAnnotatedSentence, FrameAnnotation, FrameElementAnnotation
from frame_semantic_transformer.data.augmentations import (
    DataAugmentation,
    LowercaseAugmentation,
    RemoveEndPunctuationAugmentation,
)
class SwedishTrainingLoader(TrainingLoader):
    """
    Training Loader for Swedish
    This class tells FrameSemanticTransformer how to load the
↪   Swedish FrameNet training data
    """
    train_sentences: List[FrameAnnotatedSentence]
    test_sentences: List[FrameAnnotatedSentence]
    val_sentences: List[FrameAnnotatedSentence]
    def __init__(self, swedish_framenet_xml_file, test_portion=0.1,
    ↪  val_portion=0.1, seed=42):
        # parse annotated sentences from XML
        annotated_sentences = []
        tree = ET.parse(swedish_framenet_xml_file)
        root = tree.getroot()
        for xml_frame in root.findall(".//Sense"):
            frame = extract_frame(xml_frame)
            for child in xml_frame:
                if 'example' in child.tag:
                    annotated_sentences.append(
                    extract_example(child, frame.name))
        # split into train/test/val
        random.Random(seed).shuffle(annotated_sentences)
        num_test = int(test_portion * len(annotated_sentences))
        num_val = int(val_portion * len(annotated_sentences))

        self.test_sentences = annotated_sentences[0:num_test]
        self.val_sentences = annotated_sentences[num_test:num_test +
        ↪  num_val]
        self.train_sentences = annotated_sentences[num_test +
        ↪  num_val:]

    def load_training_data(self):
        return self.train_sentences
    def load_validation_data(self):
```

```
        return self.val_sentences
    def load_test_data(self):
        return self.test_sentences
    def get_augmentations(self) -> List[DataAugmentation]:
        """
        These augmentations try to increase the training data by
↪   making safe tweaks to the text
        For instance, removing the punctuation at the end, or
↪   lowercasing the whole sentence
        """
        return [
            RemoveEndPunctuationAugmentation(0.3),
            LowercaseAugmentation(0.2),
        ]

from nltk.stem import SnowballStemmer
swedish_stemmer = SnowballStemmer("swedish")

class SwedishInferenceLoader(InferenceLoader):
    """
    Inference loader for Swedish
    This class tells FrameSemanticTransformer which frames and LUs
↪   are available during inference
    """
    frames: List[Frame]

    def __init__(self, swedish_framenet_xml_file, test_portion=0.1,
↪   val_portion=0.1, seed=42):
        # parse annotated sentences from XML
        self.frames = []
        tree = ET.parse(swedish_framenet_xml_file)
        root = tree.getroot()
        for xml_frame in root.findall(".//Sense"):
            frame = extract_frame(xml_frame)
            self.frames.append(frame)

    def load_frames(self):
        return self.frames

    def normalize_lexical_unit_text(self, lu):
        """
        This method normalizes lexical unit text for Swedish
↪   during inference
        Lexical Units help give hints to the model about what
↪   frames are likely
        """
```

```
        normalized_lu = lu.lower()
        normalized_lu = re.sub(r"\..+$", "", normalized_lu)
        normalized_lu = re.sub(r"[^a-ö ]", " ", normalized_lu)
        return "_".join([swedish_stemmer.stem(word) for word in
        ↪  normalized_lu.split()])
```

Calling the training procedure, with re-using the "train" function of the English SRL model and replacing the English data-loaders by our Swedish data-loaders, defined as above.

```
MODEL_TYPE = 'small' # or 'base'
from frame_semantic_transformer.training.train import train
train(
    # for Model 1
    base_model=f"chanind/frame-semantic-transformer-{MODEL_TYPE}",
    # for Model 2
    base_model=f"google/mt5-{MODEL_TYPE}",
    ...
    # Here is where we pass in our Swedish data-loaders
    training_loader=SwedishTrainingLoader('./swefn.xml'),
    inference_loader=SwedishInferenceLoader('./swefn.xml'),
)
```

A.2 Appendix 2: Examples from SweFN

Example annotations for frame "Placing": 4 sentences have FE "Goal" marked on a phrase that begins with word "**på**" and 2 sentences with annotation of FE "Place" on a phrase that begins with "**på**".

```
<karp:example>
      <karp:e name="Agent">Tullkvinnan</karp:e>
      <karp:e name="LU">placerade</karp:e>
      <karp:e name="Theme">väskan</karp:e>
      <karp:e name="Goal">på ett stort blankt bord</karp:e>
      <karp:text>.</karp:text>
</karp:example>
<karp:example>
      <karp:e name="Theme">Han</karp:e>
      <karp:e name="LU">stationeras</karp:e>
      <karp:e name="Goal">på finländska UD i
      Helsingfors</karp:e>
      <karp:text>.</karp:text>
</karp:example>
<karp:example>
      <karp:e name="Agent">Man</karp:e>
      <karp:e name="LU">lastade</karp:e>
```

```
      <karp:e name="Theme">sina instrument</karp:e>
      <karp:e name="Goal">på bilar och mopeder och
      cyklar</karp:e>
      <karp:text>och försvann ut i vattnet.</karp:text>
</karp:example>
<karp:example>
      <karp:e name="Agent">Han</karp:e>
      <karp:text>hade</karp:text>
      <karp:g n="2" />
      <karp:e name="LU">dukat fram</karp:e>
      <karp:g n="5" />
      <karp:e name="Goal">på köksbordet</karp:e>
      <karp:text>så vi satt och åt tillsammans hela familjen ,
      det är INTE ofta det händer !</karp:text>
</karp:example>
```

```
<karp:example>
      <karp:text>För det är tydligen</karp:text>
      <karp:e name="LU">
        <karp:e name="Time">datum</karp:e>
        <karp:g n="1" />
        <karp:e name="LU">parkering</karp:e>
      </karp:e>
      <karp:e name="Place">på Jungfruvägen</karp:e>
      <karp:text>.</karp:text>
</karp:example>
<karp:example>
          <karp:e name="Agent">Axel</karp:e>
          <karp:e name="LU">
            <karp:e name="Manner">fick</karp:e>
            <karp:g n="1" />
            <karp:e name="LU">parkerade</karp:e>
          </karp:e>
          <karp:e name="Place">på Odengatan</karp:e>
          <karp:text>, detta gjorde honom pepp .</karp:text>
</karp:example>
```