Sign Language Translation Mobile Application and Open Communications Framework

Deliverable 4.5: A hybrid intermediate representation

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101017255

Project Information
Project Number: 101017255
Project Title: SignON: Sign Language Translation Mobile Application and Open Communications Framework
Funding Scheme: H2020 ICT-57-2020
Project Start Date: January 1st 2021

Deliverable Information
Title: A hybrid intermediate representation
Work Package: WP4
Lead beneficiary: Tilburg University
Due Date: 31/12/2023
Revision Number: V0.3
Authors: Dimitar Shterionov, Bram Vanroy, Adrian Nunez Marcos, Henk van den Heuvel, Aditya Parikh, Xabier Soto
Dissemination Level: Public
Deliverable Type: Other

Overview: This document describes the development and evaluation of machine translation pipelines for translation between signed and spoken languages. These pipelines and the different components are formed by combining different models and approaches including symbolic-based, linguistically motivated approaches and embedding-based ones. This, final deliverable of WP4, covers the combined work of

different research teams and their outputs, described in other WP4 deliverables, towards the ultimate goal of automating the translation of signed and spoken languages.

Revision History

Version #	Implemented by	Revision Date	Description of changes
V0.3	Dimitar Shterionov	15/12/2023	Final version
V0.2	Dimitar Shterionov	13/12/2023	Revision after partners' reviews
V0.1	Dimitar Shterionov	08/12/2023	Initial Draft

The SignON project has received funding from the European Union's Horizon 2020 Programme under Grant Agreement No. 101017255. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SignON project or the European Commission. The European Commission is not liable for any use that may be made of the information contained therein.

Approval Procedure

Version #	Deliverable Name	Approved by	Institution	Approval Date
V0.1	D4.5	Aoife Brady	DCU	08/12/2023
V0.1	D4.5	Marco Giovanelli	FINCONS	13/12/2023
V0.1	D4.5	Vincent Vandeghinste	INT	11/12/2023
V0.1	D4.5	John O'Flaherty	MAC	08/12/2023
V0.1	D4.5	Santiago Egea Gómez	UPF	12/12/2023
V0.1	D4.5	Irene Murtagh	TU Dublin	14/12/2023
V0.2	D4.5	Joni Dambre	UGent	15/12/2023
V0.1	D4.5	Jorn Rijckaert	VGTC	08/12/2023
V0.1	D4.5	Henk van den Heuvel	RU	08/12/2023
V0.1	D4.5	Catia Cucchiarini	TaalUnie (NTU)	12/12/2023
V0.2	D.4.5	Myriam Vermeerbergen	KU Leuven	15/12/2023
V0.1	D.4.5	Rehana Omardeen	EUD	11/12/2023
V0.1	D4.5	Mirella De Sisto	TiU	11/12/2023

Acronyms

The following table provides definitions for acronyms and terms relevant to this document.

Acronym	Definition
AMR	Abstract Meaning Representation
ASR	Automatic Speech Recognition
NLP	Natural Language Processing
LSE, NGT, VGT, ISL, BSL, DSGS	Spanish, Dutch, Flemish, Irish, British and Swiss German Sign Languages (in that order)
WMT	Workshop of Machine Translation (While the abbreviation is still remaining for historical reasons, the workshop has turned into an international conference)
SLMT	Sign Language Machine Translation
SLR	Sign Language Recognition
Term	**Definition**
Interlingua	An intermediate language representation
mBART	A multilingual language model
SignON InterL, InterL-E, InterL-S	Specific interlingua representations developed within SignON - jointly referred to as InterL, InterL-E - embedding-based interlingua, InterL-S symbolic interlingua
SignNets	Semantic networks of signs, similar to WordNets (for words).
NGT HoReCo	A new data set of aligned signed and spoken data.
BLEU, ROUGE, CHRF, BLEURT	Comparison-based metrics for the evaluation of the quality of generated text.

Table of Contents

1. Introduction

This deliverable describes the final status in the development of pipelines for translation using the intermediate representation – InterL. It covers the work we conducted with different models and model combinations.

The purpose of the interlingual representation is two-fold: (i) to benefit from the translation coverage and robustness of large multi-language models and (ii) to reduce the number of language-specific models. With respect to (ii) if we "simply" build recognition + translation models for each individual language combination (English, Spanish, Irish, Dutch (Netherlands), Dutch (Flanders) in both text and audio and the sign languages LSE, NGT, VGT, ISL, BSL) we would require to develop and, most importantly, deploy in our framework **210 individual** models.

In this deliverable we report our qualitative results and findings with respect to the InterL, which encapsulates the InterL-E and the InterL-S together, and the combination with other models and pipelines. Then we suggest potential research and development directions for future work beyond the span of the project.

2. InterLingua

In this section we present an overview of the InterL with a summary of the work conducted on its two underlying forms - the embedding based (InterL-E, see Section 2.1) and the symbolic (InterL-S, see Section 2.2). We focus on their combination and the use of other models to conduct translation between the different languages and modalities.

2.1. InterL-E

At the beginning of the project we looked at existing embedding models and selected a suitable one to adopt as our intermediate vectorial / embedding-based representation. In particular, we selected mBART (Liu et al. 2020) as defined in deliverable *D4.3 First distributional intermediate representation based on embeddings*. We then extended it to support the encoding and decoding of both signed and spoken language representations, as shown in *D4.4 Second distributional intermediate representation based on embeddings - InterL-E*.

Supporting sign language is done via aligning SLR model embeddings with those of the InterL-E. Up until month 11[1] of the project, the SLR and the InterL-E development were conducted in parallel and independently. After that a joint plan was drawn towards the aforementioned approach, which was first applied with the model from (Camgoz et al., 2020) and showed promising results (by month 25). After the first proof-of-concepts, we commenced technical work on integrating those two in a sign-to-text pipeline. We experimented with different options and pipelines, e.g. fine-tuning the SLR and InterL-E models jointly (partially freezing the translation model in the case of mBART pipeline, see below for details), using different embeddings from the SLR, etc. These pipelines along with empirical evaluation are discussed in Section 3.2.

When it comes to text-to-sign, i.e. generating signed utterances via our 3D virtual signer (for the final updates see deliverable *D1.12 User generated data and D5.6 Final Sign language-specific lexicon and structure*) we utilise both, a symbolic approach (InterL-S) and an embedding-based approach (InterL-E). In other words, we use embedding based methods to generate symbolic representations. In a nutshell, an mBART model[2] (embedding-based representation) has been finetuned to automatically generate symbolic representations, specifically Abstract Meaning Representation (AMR) graphs, for English, Spanish and Dutch which, after post-processing and reverse look-ups in a modified Signbank, yield a sequence of glosses. For an overview of this approach, see *D4.2 Second symbolic intermediate representation - InterL-S*. These are then used as input to the planner of the virtual signer.[3]

Supporting spoken language. The speech-to-text pipeline first converts the speech input into text which is then translated through the InterL-E. The text-to-speech pipeline sends text to either an external text-to-speech[4] or framework provided by www.acapela-group.com or to the text-to-speech system provided by the OS of the mobile device.[5] We present details and evaluations in Section 3.4.

[1] By month 23 models for VGT & NGT we complete, and by month 27 the models for ISL & BSL. Final versions of these 4 languages (small improvement) were deployed in M29. First language specific version for LSE was released in month 35 (after data had been collected – see D7.4 for details and timeline).

[2] After comparing the mBART model used for the InterL-E (50-large) to other versions we decided to use cc25 which performed best for the AMR task.

[3] The virtual signer builds on an elaborate framework and supports multiple forms of inputs.

[4] Both Google's text-to-speech and iOS's text-to-speech frameworks (depending on the platform the app runs on) could also be used.

[5] This allows us or anyone who uses the SignON framework to seamlessly connect the output of the InterL with a preferred text-to-speech system. This is also possible with the speech-to-text component; however, the models we developed have been fine-tuned to specific languages, use-cases and types of speech (e.g. atypical speech).

2.2. InterL-S

We started the InterL-S exploration with three different approaches which are summarised in *D4.1 First symbolic intermediate representation - InterL-S*. These include (i) a rule-based system for generating glosses from spoken language, (ii) WordNets and SignNets and (iii) Abstract Meaning Representation (AMR). While we could add the work on Sign_A to this set, it is more associated with the sign synthesis side, rather than the InterL (see *D5.4 First Sign language-specific lexicon and structure*, *D5.5 Second Sign language-specific lexicon and structure* and *D5.6: Final Sign language-specific lexicon and structure* for details). Following the initial work on these three approaches, we then focused on the hybrid approach of text-to-AMR(-to-gloss-to-Avatar). As discussed in *D5.5 Second Sign language-specific lexicon and structure* the reasons for the raised attention to AMR was the ease of automation which allows us to not only deploy text-to-AMR models along with the InterL-E but also to cover multiple source languages. This was also augmented by the decision to rely on glosses for the synthesis, as explained further in *D5.6: Final Sign language-specific lexicon and structure*.

In parallel, the work on SignNets and NLP was progressing. As defined in Schuurman et al (2023), similar to a wordnet, which is a large semantic network stored in a database, a SignNet is a semantic network that includes various types of data available for a specific SL, "*also signs (and images/videos showing them), with their phonological elements, like hand shapes, position, orientation, glosses, their phonetic transcriptions, examples of use (in both the SL concerned and the surrounding spoken language), definitions and identifiers of entries in corpora where some attestations of the signs can be found, etc. This would mean that SignNets are semantic networks on their own, applied to visual-gestural data, and containing links to wordnets, rather than being integrated in those.*"

As SignNets are linguistically-motivated representations of sign languages the work we conducted on this topic is detailed in *D3.3 Linguistic description for ISL, BSL, VGT, NGT and LSE*. Our work on the NLP pipelines is detailed in *D3.6 Second Natural Language Processing pipeline* with some experimental work, in the context of translation, presented in Section 3.3 of this deliverable.

3. Quality Assessment

In this section, we present the evaluation of the pipelines for sign-to-text, speech-to-text and text-to-sign. These pipelines use the InterL as intermediate representation or build on additional/other models. A reflection on the InterL is given in Section 4.

3.1. NGT HoReCo as common testing data

In 2023, we developed a new corpus with data from the hospitality domain. This domain was indicated by the deaf participants in our co-creation events (WP1) as the one where they would most likely use the SignON application and thus we started collecting data for this domain to cover the data gap. The NGT HoReCo (De Sisto et al., 2023) started as a multimodal parallel corpus of Dutch and Sign Language of the Netherlands (NGT). 297 hotel reviews in written Dutch[6] were translated into NGT videos by in total 6 professional, deaf translators. Each review was translated by a single translator. The length of the Dutch reviews varied from around 15 to 400 words; the NGT video duration ranged from around 10 seconds to around 4 minutes. The total amount of words contained in the corpus is 21,825; the NGT translations consist of over 3.5 hours of videos.[7] While NGT HoReCo started as a English-Dutch-NGT corpus, we took the initiative to translate it into other languages. First it was translated into VGT, recently also into Irish (GA) and Spanish, and translation into LSE is ongoing, with BSL and ISL for future work. The NGT HoReCo gave us the opportunity to have a common, domain-specific, unified corpus. However, it lacked speech data. For every spoken language native speakers were tasked to record utterances from the corpus. For the purpose of a common testbed to evaluate the recognition and translation pipeline, audio recordings were made of the 15 shortest English contributions to the full HoReCo dataset. These 15 sentences were translated into Spanish, Dutch (Netherlandic + Belgian variants), and Irish, and read out by 30 speakers. The table below shows an overview of the speakers.

[6] The Dutch text is a translation of the original reviews written in English. The translation is done by a professional translation company.
[7] Source: https://taalmaterialen.ivdnt.org/download/ngt_horeco1_0/

	Dutch - NL	Dutch - BE	English	Spanish	Irish	All
Male	10	2	0	3	1	16
Female	5	2	2	2*	3	14
All	15**	4***	2	5	4	30

* One participant is non-native, fluent speaker.

** One participant is deaf using their voice.

*** One participant is hard of hearing.

3.2. Sign to text

In this section, we show the results of the sign-to-text translation from the 5 sign languages (VGT, NGT, ISL, LSE and BSL) into the 4 spoken languages (Dutch, Irish, English and Spanish) of the project, i.e. their orthographic representations. Additionally, we also include the results on the DSGS-Deutsch language pair part of the WMT SLT shared task (Müller et al., 2023). In Section 3.2.1 we present the summary of the datasets employed for the evaluation, in Section 3.2.2 we describe the models we explored for these experiments, in Section 3.2.3 we present the evaluation metrics used for the evaluation and, finally, in the summary in Section 3.2.4 we show the best configurations and hyperparameters.

3.2.1 Data

The table below shows a summary of the datasets employed for sign-to-text and their train/dev/test distribution:

Dataset	Source Lang	Target Lang	Train	Dev	Test
VRT-News	VGT	Dutch	5,290	500	500
Corpus NGT	NGT	Dutch	18,155	1,799	2,187
Signs Of Ireland	ISL	Irish / English	0*	0*	15
BSL Corpus	BSL	English	14,732	500	500
LSE_eSaude_UVIGO	LSE	Spanish	4,050	300	300
WMT	DSGS	Deutsch	8,044	500	496

** There are only 15 aligned samples so these were used for the test, as they are not enough to train a model.*

The train/dev/test split for the NGT Corpus was decided based on the signers. We used the same signer distribution as the one used to train the models of *D3.2 "Sign language recognition component and models"*. For the rest of the datasets, we used a random distribution taking into account the number of samples to select an appropriate dev and test set sizes. The Signs of Ireland dataset is very small, so all the data is used for the test set (hence no model will be trained using this dataset). Regarding WMT, we extracted the dev set randomly from the train set, and we used the test set as provided by the organisers, showing also the results for both separate sources Signsuisse (SS) and SRF.

The Content4All's VRT-News dataset was collected in (Camgöz et al., 2021); LSE_eSaude_UVIGO was first presented in the work of Docío-Fernández et al., 2020; Corpus NGT, available at www.corpusngt.nl, was introduced in (Crasborn, et al. 2008); Signs of Ireland was presented in (Leeson and Nolan, 2008); the BSL Corpus in (Schembri et al., 2011) and, finally, the WMT shared task can be found in the following link: https://www.wmt-slt.com/.

3.2.2 Models

The first pipeline we experimented with was based on the mBART model underlying our InterL-E. After experimenting with the model, we also conducted tests with other models. These are summarised below:

(1) mBART

mBART (Liu et al. 2020) is InterL-E's foundational model. It is a text-to-text model based on transformers, pre-trained for denoising corrupted text and, afterwards, fine-tuned to translate between 50 different languages (including Dutch, English and Spanish). In our case, to adapt it to the sign-to-text task, since the input will be SLR vectors/embeddings, we modified it to accept embeddings as inputs as in *D4.4 "Second distributional intermediate representation based on embeddings - InterL-E"*. The SLR component from *D3.2 "Sign language recognition component and models"* is used to extract the feature vectors/embeddings that are used as input to the mBART encoder. These embeddings can be extracted from two points of the SLR neural network, hence obtaining two types of input embeddings. We refer to them as "spatial" or "temporal" embeddings from the SLR component. There are also two versions of the SLR component, one with hidden size 128 and another one with hidden size 192. The SLR hidden size will be specified in the experiments.

Due to the difference in the hidden size between the SLR component (128 or 192) and mBART (1,024), a linear layer to upsample the dimension and (optionally) a normalisation layer are used.

Besides, this module can be either fully fine-tuned or only fine-tuned using adapters (Houlsby et al., 2019). By using adapters, a few extra weights are added to the network and only those are fine-tuned while the rest of the network remains untouched.

(2) 3-layer transformer

The second type of model is also a transformer (Vaswani et al 2017) model but with randomly initialised weights (no pre-training). Its architecture is also smaller than mBART: it has a hidden size of 512, the number of heads is 8 and the fully-connected layer dimension is 2048. Once again, it takes SLR embeddings in the exact same way the previous model does. In contrast to mBART, this transformer needs to always be fully trained for our task.

(3) SLR component + mBART

Similar to the first model, in this case the SLR component is also fine-tuned alongside mBART. Hence, the gradients flow through both models. The SLR component's embeddings are sent to a linear layer to upsample their dimensionality (to match mBART's hidden size) and then to mBART as input just like in the first model.

(4) SLR component + 3-layer transformer

This model uses the architecture of the previous model (i.e. SLR component and a SLT model fine-tuned together) but in this case the SLT model is not mBART, instead we employed the transformer presented as model (2). The size of the model is also the same.

3.2.3 Measurements

The following metrics are used to evaluate the performance of sign-to-text models. The evaluation results are shown in Section 3.1.4:

(1) **BLEU-1/2/3/4** (Papineni et al., 2022): N-grams of the hypothesis are compared with the N-grams of the reference and precision of n-grams is computed (position independent matches) and

normalised between 0 and 100. BLEU-1 is computed using only 1-grams, BLEU-2 combines 1-grams and 2-grams precision and so on until BLEU-4.

(2) ROUGE (Lin, 2004): Rouge-L (Recall-Oriented Understudy for Gisting Evaluation) is based on the longest common subsequence (not necessarily consecutive, but in order) between the hypothesis and the reference.

(3) CHRF (Popović, 2015): The CHRF (CHaRacter-level F-score) measures the similarity between the hypothesis and the reference computing the F-score at character-level n-grams.

3.2.4 Experiments (to regional spoken language, written form)

Each of the following tables shows the results for each sign language of the project and its corresponding regional spoken language. Only the best experiments are shown, alongside the hyperparameter set employed and the discarded values that were also explored. For the ISL-English experiments, since there is no data to train a model, the model employed was the BSL-English model. Note that we do not do ISL-Irish translation since we machine translate (using the application's text-to-text mBART) the ISL-English model's output to Irish. Irish is out of the original mBART model and would be problematic to fine-tune a specific model to include Irish for sign-to-text translation.

NGT - Dutch / Netherlands

Model	*Hyperparams*	*BLEU-1*	*BLEU-2*	*BLEU-3*	*BLEU-4*	*CHRF*	*ROUGE*
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*6.57*	*2.46*	*1.20*	*0.61*	*12.73*	*7.59*
Other hyperparameters that were tried but the obtained results were lower: (1) not using adapters, (2) translation beam alphas -1, 1 and 2, (3) patience 20, (4) learning rates *1e^{-3}* and *1e^{-5}*, *(5) not subsampled sequences. (6) stop metric chrF, translation loss, perplexity, (6) weight decay 0.01, (7) dropout 0.2. The best results were obtained by mBART.*							

VGT - Dutch-BE

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *TargetLanguages=Eng,Dut,Es*	*10.43*	*3.01*	*1.15*	*0.44*	*16.5*	*8.94*
Other hyperparameters that were tried but the obtained results were lower: (1) not using adapters, (2) learning rates *1e^{-3}*, *5e^{-4}*, *1e^{-4}*, *5e^{-5}*, *5e^{-6}* and *1e^{-6}*, (3) stop metric translation loss, (4) temporal features, (5) normalised input features, (6) including mBART's original EOS and English SRC_LANG tokens in the encoder, (7) translation beam size for the development set to 3, (8) stacking feature vectors to match the number of dimensions of mBART's hidden size, (9) using only Dutch as target language. *The best results were obtained by mBART.*						

LSE - Spanish

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*3.5*	*0.73*	*0.32*	*0.00*	*12.43*	*3.17*

BSL - English

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*4.85*	*1.10*	*0.33*	*0.00*	*8.69*	*4.63*

ISL - English

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	-	*5.07*	*1.36*	*0.49*	*0.20*	*8.82*	*5.02*
Since very few samples were available, the best BSL-English model presented above was employed to evaluate the ISL-English translation task.							

DSGS - Deutsch (WMT_all)

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*4.03*	*0.88*	*0.00*	*0.00*	*13.62*	*4.01*

DSGS - Deutsch (WMT_SS)

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*4.65*	*1.09*	*0.00*	*0.00*	*14.72*	*5.03*

DSGS - Deutsch (WMT_SRF)

Model	Hyperparams	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CHRF	ROUGE
mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *Learning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=3* *Dropout=0.1* *Weight decay=0.001*	*3.61*	*0.36*	*0.00*	*0.00*	*13.41*	*3.70*

The results shown here are comparable to those shown in *D4.4 "Second distributional intermediate representation based on embeddings - InterL-E"*, where the sign-to-text model was evaluated on VGT-Dutch using the VRT-News dataset. For LSE-Spanish, BSL-English, ISL-English and DSGS-Deutsch (WMT), we used the hyperparameters that obtained best results for NGT-Dutch, so the results are better for the latter.

Regarding WMT (Müller et al., 2023), they measured BLEU-4, CHRF and BLEURT, so for a better comparison we calculate these metrics on the whole test set using the same configuration used in WMT[8]. These results are shown in the following table, including the WMT baseline scores for reference:

DSGS - Deutsch (WMT_all) [same metrics configuration]

System	*BLEU-4*	*CHRF*	*BLEURT*
WMT baseline	*0.09*	*12.4*	*0.072*
Our system	*0.11*	*13.6*	*0.246*

In the case of text-based metrics, our BLEU-4 and CHRF results are above the baselines presented in the WMT SLT shared task (Müller et al., 2023), even if we optimised the hyperparameters for another language pair. In the case of the most significant neural metric BLEURT, our system would have been the best submission by a slight difference (0.246 Vs. 0.243). This means we were able to achieve results comparable to the state of the art, even though the results are far from being useful for a real-world application.

These low results may indicate that the model is not capable of learning from the data. To test this, we made various experiments using a subset of the training set for the training, development and evaluation steps. We also removed all the regularisation (weight decay and dropout). With this setting, the model should be capable of memorising the input data. We tried to repeat this for mBART and the smaller transformer model, using training set subsets of size 10, 100, 1,000, 10,000 and the full training set. Both types of models were able to memorise 10 and 100 samples. However, the small transformer obtained a BLEU <50 for 1,000 samples and struggled to memorise the 10,000 subset, obtaining results ranging from 1 to 30 BLEU. mBART was able to correctly memorise the training set but failed to generalise to the

[8] BLEU: nrefs:1|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.3.1|mateo:1.1.3
chrF2: nrefs:1|bs:1000|seed:12345|case:mixed|eff:yes|nc:6|nw:0|space:no|version:2.3.1|mateo:1.1.3
bleurt: nrefs:1|bs:1000|seed:12345|c:BLEURT-20|version:commit cebe7e6|mateo:1.1.3

development and evaluation sets, hallucinating sentences that had no relation with the input utterance. This leads to higher CHRF and ROUGE values in contrast to smaller, not-pretrained transformers.
For the full list of experiments, see the Annex A “Sign-to-text experiments”.

3.3. Using NLP pipelines

In this section, we present quality metrics on the NLU pipeline presented in D3.6 “Second Natural Language processing pipeline”. As we do not have any annotated corpus to measure the performances of this pipeline, we conducted a manual evaluation using 10 samples extracted from NGT HoReCo dataset. The NLU pipeline includes the following process: (1) Text Normalisation, (2) Linguistic Feature Tagging and (3) Word Sense Disambiguation. In order to assess (1), we manually introduce noise in the form of typos in the input text and count how many of those alterations were corrected. In the instance of (2), we used well-established models implemented in Spacy[9]; it is reported high performances for the language covered in its webpage. Finally, we manually validate the outputs generated by our Word-Sense Disambiguation module. In the case of Irish, we did not find a suitable resource to perform (3) for this language. Below, we present the results for the nlp pipeline implemented.

Language	***Text Normalisation***	***Linguistic Tagging[10]***	***Word-Sense Disambiguation***
English	*11 of a total of 16 alterations were corrected*	*https://spacy.io/models/en*	*30 of 36 terms were correctly disambiguated*
Spanish	*7 of a total of 13 alterations were corrected*	*https://spacy.io/models/es*	*13 of 39 terms were correctly disambiguated*
Dutch	*11 of a total of 13 alterations were corrected*	*https://spacy.io/models/nl*	*15 of 38 terms were correctly disambiguated*
Irish	*6 of a total of 18 alterations were corrected*	*https://stanfordnlp.github.io/stanza/performance.html*	*No implemented*

The errors in the Text Normalisation phase normally are caused by the lack of context for the misspelling checking. These errors may be propagated to the next steps in the pipeline producing more errors in the subsequent processes. We can observe that the language with the best results is English, while the performances observed for the rest are positive, but very improvable. Although the results in Word-Sense Disambiguation are limited for Spanish and Dutch, many terms were related to close meanings, but not the exact.

[9] https://spacy.io
[10] In the case of spacy models, the performances of each model are reported in section “Accuracy Evaluation”

3.4. Speech to text

In this section, we present the results of the automatic speech recognition component that transcribes oral messages into text. Spoken language input is possible in four languages: English, Spanish, Irish, and Dutch (Belgian variant), see above. The ASR functionality supports both Northern and Southern Dutch accents. In section 3.4.1, we provide a summary of the training datasets for ASR models in different languages. The section 3.4.2 offers a brief description of the various ASR models we trained, fine-tuned and used as it is. Finally, the ASR performance across audio recordings of 30 speakers from the HoReCo dataset is presented in section 3.4.3.

3.4.1. Training Dataset

Language	Data resources
Dutch	- Audio Data: 483 hours of speech from Northern Dutch speakers in the Spoken Dutch Corpus (CGN). - Text Data: 530 million words from CGN and the Twente Nieuws Corpus. - Lexicons: 181k Dutch words derived from the CGN lexicon.
Irish	- Audio Data: We used all validated utterances from the Common Voice Irish dataset. An additional hour of Irish speech data was derived from the Living Audio dataset. We also incorporated all Irish utterances from the Google Fleurs dataset. In total, we had 9,274 Irish utterances, equivalent to 13.5 hours of speech. - Text Data: CC-100: Monolingual datasets from Web Crawl Data, covering over 100 languages, including Irish. It contains 84 million word tokens of Irish of which 0.12 million word types with frequencies higher than 10. - Lexicons: G2P model based on 13,300 seed Irish pronunciations acquired from Wikipron.
English	- Audio Data: We used the GigaSpeech corpus with a total of 10,000 hours of transcribed audio data. For the Acoustic model (AM) we used primarily the 'S' subset of GigaSpeech corpus with 250 hours of audio data for training. The 'S' subset contains 230,068 utterances.

	- Text Data: The textual data sourced from the "Google One Billion Word Benchmark" text corpus was used. It contains nearly one billion word tokens; it includes 400k word types with a frequency of more than 10. - Lexicons: Pronunciation lexicons for English used in the ASR system were trained based on CMUdict – an English pronunciation lexicon with 135k entries
Spanish	- Audio Data: The Common Voice Spanish dataset with 213,244 utterances, equivalent to 313.56 hours of speech was used. - Text Data: We used the Spanish Billion Words Corpus with nearly 1.5 billion Spanish tokens and 0.54 million word types with frequency higher than 10. - Lexicons: The lexicons are generated using a G2P tool based on SAMPA (Speech Assessment Methods Phonetic Alphabet).

3.4.2. Models

(1) Kaldi

In the SignON application, initially all of the ASR models used for English, Spanish, Dutch, and Irish languages were created using a classical modular/hybrid ASR approach with the Kaldi toolkit[11]. This approach breaks down the overall ASR architecture into smaller functional components: acoustic model (AM), language model (LM), and pronunciation lexicon. The modular approach allows us to train our own AM, and LM with a task-dependent customised vocabulary. The use of this modular approach has its advantages, especially for specific domains like different languages, dialects or speakers. All of these hybrid ASR models were deployed via a web service[12] and integrated with the SignON mobile application framework as version 1 (V1). However, in the later stage end-to-end models showed significantly higher performance and in the newer version of ASR we utilised end-to-end models like wav2vec 2.0 and whisper ASR. Currently, Irish ASR is served via the Kaldi hybrid approach.

(2) Wav2vec 2.0

[11] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Vesely, K. (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding* (No. CONF). IEEE Signal Processing Society.

[12] https://restasr.cls.ru.nl/api-docs/

In version 2 (V2) of the ASR models, we employed an end-to-end ASR approach. Wav2vec 2.0 is a state-of-the-art self-supervised learning framework for speech processing, specifically designed for ASR. It has demonstrated an impressive performance in ASR. We utilised a publicly released pre-trained wav2vec 2.0 model, XLS-R[13], which was trained on 436k hours of publicly available speech audio and fine-tuned them with our data. This model is available on Hugging Face. The largest variant has up to 2 billion parameters[14]; intermediate checkpoints having 300 million parameters[15] and 1 billion parameters[16]. Dutch/Flemish ASR and Spanish ASR is served via wav2vec 2.0 XLS-R models. Dutch/Flemish ASR is fine-tuned with a 2B-parameter model whereas Spanish ASR is fine-tuned with a 300M-parameter model.

(3) Whisper ASR

Whisper[17] is an alternative to the Kaldi and wav2vec 2.0 models. It is an ASR system trained on 680,000 hours of multilingual and multitask[18] supervised data collected from the web. This model has shown that the use of such a large and diverse dataset leads to improved robustness with accents, background noise and technical language. Moreover, it enables transcription in multiple languages including English, Spanish and Dutch (but not Irish), which are target languages in the SignON project. The Whisper ASR models[19] are available in five different variants, each varying in size. These variants range from tiny (39M parameters) and base (74M parameters) to small (244M parameters), medium (769M parameters), and large (1550M parameters). The accuracy of the models increases as the number of parameters grows, enabling improved performance. In our experiments, we have observed that using Whisper models can achieve nearly real-time speech recognition using their different size of models and that can be very beneficial for the hearing users' ASR system. In our evaluation, we utilise the Whisper models without conducting any finetuning. We directly employ the available pretrained models, utilising them as they are, without making any modifications to their architecture or parameters. We have utilised Whisper's medium sized model (769M parameters) for English ASR.

[13] Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., Auli, M. (2022) XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale. Proc. Interspeech 2022, 2278-2282, doi: 10.21437/Interspeech.2022-143

[14] https://huggingface.co/facebook/wav2vec2-xls-r-2b

[15] https://huggingface.co/facebook/wav2vec2-xls-r-300m

[16] https://huggingface.co/facebook/wav2vec2-xls-r-1b

[17] Radford, Alec, et al. "Robust speech recognition via large-scale weak supervision." arXiv preprint arXiv:2212.04356 (2022).

[18] In whisper models, multitask refers to its ability to perform tasks such as automatic speech recognition, voice activity detection, and any-to-English speech translation.

[19] https://github.com/openai/whisper

3.4.3 Results over HoReCo Audio Recordings

In the table below, we present the performance of ASRs over the audio recordings of 30 test speakers in the hospitality domain. The reported Word Error Rate (WER) scores align well with the targeted values outlined in the SignON project proposal. However, an exception is observed for the Irish language in the HoReCo use-case task, where the system for two out of four speakers performed notably worse than the others. It is important to note that although the recorded speakers in this testset were Irish, they were not native speakers of the Irish language.

The performance of the Speech Recognition component on the HoReCo test set refers to the audio recordings of 30 test speakers in the hospitality domain are provided in the table below:

Language	ASR V1 WER	ASR V2 WER
English	24.23%	5.15%
Spanish	12.82%	9.48%
Dutch (Netherlands)	15.61%	5.23%
Dutch (Flanders, Belgium)	20.29%	7.79%
Irish	39.46%	42.77%

Based on these results, in the SignON mobile application, English ASR is served via the Whisper medium model. Spanish and Dutch-Flemish ASR are served via the wav2vec 2.0 XLS-R model, while Irish is served via a Kaldi-based hybrid ASR model.[20]

For the translation, the mBART model included in the app was used. This was described in *D4.11 "First Adaptable Pipeline for Training and Updating the InterL".*

3.4.3. Experiments to regional languages

The following table summarises the results of the audio-to-text pipeline. This includes the ASR process to obtain a transcripted text and the translation of that text (in a given source language) to another target language. In this case, the target language is the surrounding regional spoken language of each sign language.

[20] It is worth noting that in our application, the ASR V1 (that is, the previous version) uses only Kaldi models.

Source Language	Target Language	BLEU-1	BLEU-2	BLEU-3	BLEU-4	TER	METEOR
English	Dutch	0.79	0.66	0.61	0.57	0.73	0.49
English	Spanish	0.80	0.59	0.48	0.41	0.76	0.34
English	Irish	0.87	0.71	0.60	0.54	0.82	0.39
Irish	Spanish	0.63	0.24	0.07	0.03	1.39	0.02
Spanish	Irish	0.72	0.28	0.07	0.01	1.25	0.01
Dutch	Spanish	0.68	0.26	0.10	0.05	2.62	0.01
Spanish	Dutch	0.60	0.27	0.12	0.09	1.87	0.05
Dutch	Irish	0.71	0.26	0.06	0.01	1.26	0.01
Irish	Dutch	0.64	0.28	0.13	0.08	1.33	0.05

3.4.4. Experiments to English

The following table summarises the results of the audio-to-text pipeline. This includes the ASR process to obtain a transcripted text and the translation of that text (in a given source language) to another target language. In this case, the target language is always English.

Source Language	Target Language	BLEU-1	BLEU-2	BLEU-3	BLEU-4	TER	METEOR
Dutch	English	0.73	0.59	0.53	0.49	0.91	0.48
Spanish	English	0.61	0.43	0.35	0.29	2.11	0.33
Irish	English	0.65	0.42	0.32	0.27	1.24	0.3

3.5. Text-to-sign

3.5.1. Text to gloss

To bridge the gap between written input text and synthesis by an avatar, the project has investigated a number of approaches. In this section we describe the final implementation and evaluation of the currently implemented methods. Data scarcity has necessitated a non-end-to-end solution to the problem of driving an avatar from text input. We have therefore developed two approaches of converting text into glosses, and (as the next section will show) those glosses will be used for avatar synthesis. We first describe the two approaches, and then describe the evaluation experiment and the results. Both approaches are available via a FastAPI endpoint.[21]

[21] https://github.com/signon-project-wp4/text2gloss

Rule-based text-to-gloss

A first approach is a rule-based transformation of the input text. This was described in *D4.1 "First symbolic intermediate representation - InterL-S"*. A linguistically informed pipeline was developed that can convert a given Dutch sentence into VGT glosses. An example is shown below where different steps are taken to generate the glosses (F). First, words are lemmatised (B), then rules are successively applied to decide which lemmas to delete (not relevant for glossing) and how to reorder glosses within a phrase. These rules are specific to part-of-speech tags (C), the head of a dependency relation (D) and the dependency label (E; no rule to apply in this example).

(A) Hij weet niet wat erin zit . (He does not know what is in it.)
(B) HIJ WETEN NIET WAT ERIN ZITTEN .
(C) WG-3 WETEN NIET WAT IN delete delete
(D) WG-3 WETEN-NIET WAT IN
(F) WETEN-NIET WG-3 WAT IN

The implemented pipeline, available in the API, is restricted to VGT. See deliverable D5.6 for details.

A rule-based gloss system always stays very close to the lexical representation of Dutch, as it uses lemmatisation to generate glosses. Secondly, the current implementation requires Dutch as input text. Thirdly, the glosses that are produced are "synthetic" or "pseudo-glosses", in the sense that they are automatically derived. There is no automatic check that these glosses are real, established glosses that are used in corpora or present in dictionaries or signbanks. Our second approach, based on abstract meaning representations (AMR), aims to overcome those issues.

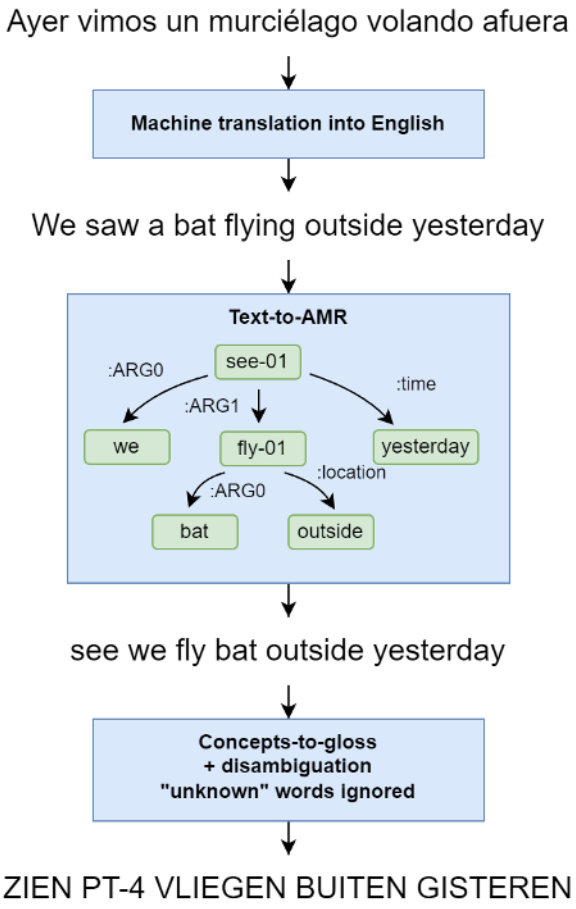
AMR-based text-to-gloss

AMR (Banarescu et al., 2013) is a semantic framework to represent meaning in a graph structure. Semantic structures are abstract and not tied to lexical surface form. In the graph, semantic relations between concepts are expressed. An example will be given below. Importantly, the concepts that are generated in AMR are always in English. These are linked to semantic frames, and the general agreement in this research is that those semantic fields are applicable across languages. In other words, the English concepts should merely be seen as an identifier of a language-independent semantic field.

AMR was first suggested as an interlingua and even as a potential complement to Sign_A, which could then be fed to the avatar. See for instance deliverables *D4.1 "First symbolic intermediate representation - InterL-S"* and *D5.5 "Second sign language-specific lexicon and structure"*. Over time, the role of AMR has changed in the project until finally it is being used as a second approach to generate glosses with the intent to overcome the issues of a rule-based approach, described above. The text-to-AMR-to-gloss pipeline is described in detail in deliverable *D4.2 "Second symbolic intermediate representation - InterL-S"*. We have trained our own text-to-AMR models (both multilingual and monolingual, supporting Dutch, English and Spanish - not Irish because it was not part of the mBART base model), which are publicly available alongside a demo.[22]

We found that the English-to-AMR model performs a lot better than multilingual models or language-specific models for the other languages, so in the current SignON implementation the text is first translated to English with the MT system that is already in place for text-to-text translation. Then, that English text is submitted to the text-to-AMR system. We finally extract semantic concepts from the AMR graph (green boxes in example below). One of the goals of this approach was to be able to constrain our vocabulary to only glosses that we can support. Therefore, the next step involved the VGT and NGT Signbanks. We modified those to include English translations of the glosses in addition to the already present Dutch translations (for more detail about this process, see D4.2.) This finally allows us to use the generalised semantic concepts that we extracted from AMR in a reverse look-up in these modified signbanks to retrieve either NGT or VGT glosses.

[22] https://huggingface.co/collections/BramVanroy/multilingual-text-to-amr-650b0fd576856b9acb257535



The pipeline of going from the Spanish sentence “Ayer vimos un murciélago volando afuera” into NGT glosses. The green boxes in AMR indicate the events and concepts that we extract.

While this pipeline works for NGT and VGT, it is not implemented for other sign languages. Word or phrase reordering has also not been implemented yet, which unfortunately has not been feasible within the time frame because we wanted to focus on a final evaluation instead. In summary, this approach allows representations that are not lexically bound but semantically derived to be used when looking up glosses from a closed vocabulary.

Evaluation

To evaluate the two approaches above, we started from the NGT Corpus which has parallel Dutch sentences and glosses. We evaluate both approaches in an “optimal” scenario where the reference glosses only contain glosses that are supported by our system and the avatar, which yields a test set of 199 sentences. With the Dutch sentences as input, we generate gloss predictions. We then make use of a series of machine translation evaluation metrics to evaluate the two approaches. Note that the rule-based approach does not support NGT, so we are evaluating the system on generating NGT. We

chose NGT because that is the only language that is currently supported by the avatar and because we found that there is a significant overlap in the vocabularies, so we hypothesised competitive performance with the AMR approach.

	BLEURT	COMET	ChrF	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Rule-based	**29.1 ± 2.4**	**60.2 ± 1.6**	**37.8 ± 2.7**	**27.9 ± 2.9**	**11.6 ± 2.5**	**5.8 ± 2.0**	**3.2 ± 1.6**
AMR-based	22.2 ± 2.0	54.0 ± 1.3	27.0 ± 2.2	20.3 ± 2.3	6.4 ± 1.6	1.8 ± 0.9	0.8 ± 0.4

Results (mean ± 95% confidence interval, with bootstrapping n=1000) of different glossing approaches on a subset (199 sents.) of the NGT Corpus. All differences are significant with $p<0.05$. For all metrics: higher is better.

Across the board we see that the rule-based system is more accurate on this dataset. Upon closer inspection of the results we find a number of explanations. Importantly, the text-to-AMR pipeline is long: it requires automatic translation to English, semantic parsing, concept extraction, and disambiguation in a constrained lexicon through concept translation. At every step extra noise is introduced that can propagate and lead to errors. The rule-based system, on the other hand, always sticks close to the source text in terms of the lexical surface forms (through lemmatisation). While we hypothesised that that is a disadvantage, it seems that it actually leads to better results than the alternative. A second reason for the performance difference is the limited scope of the Signbank. While our version contains 4,370 glosses (which includes gloss variants), many of them are linked to only one Dutch meaning. As described in D4.2 and alluded to above, those Dutch meanings and the original gloss were used to augment the Signbank with English synonyms (through multilingual WordNet as well as with gpt-3.5), so that we could go from an English word to an NGT gloss. Augmenting the Signbank in this way still has a limited coverage of the English language, i.e., there will be many English words that are not linked to an NGT gloss. Worse, however, is that - despite best efforts in disambiguation with vectorial similarity measures - an English word may also be linked to an incorrect NGT gloss.

So while the AMR pipeline has conceptual merit in its abstract meaning, not being attached to lexical form, and compatible with a closed vocabulary, those also seem to be the exact causes for the rule-based system to perform better in our evaluation. Going forward we believe that a combination of our methods is most suitable: on the one hand the relative simplicity of the rule-based pipeline and its adherence to the source text is powerful. On the other hand, we would still need to make sure that

generating those synthetic glosses are constrainable to a closed lexicon. So applying some kind of post-processing via a WordNet fallback or semantic similarity querying is necessary.

3.5.2. Glosses to avatar synthesis

As explained in other deliverables (*D1.6 "Quality Assessment Report"*, *D1.12 "User-generated data"*), the correct way to synthesise sign language is from a phonetic representation. SiGML (as a derivation from earlier sign representations such as Stokoe notation and HamNoSys) has been previously researched and extended thanks to the manual creation of use-case-specific repositories of signs in different projects. Some of them cover the healthcare domain in NGT (Esselink et al., 2022) and in LSF-CH (Strasly et al., 2018) as well as the use-case of providing railway travel announcements in NGT (Van Gemert et al., 2022). These corpora are used to map from a gloss input into a phonetic representation that contains grammatical information of signs. Afterwards, thanks to the compatibility and similarities between SigML and BML both being XML-like formats, including MFs, NMFs, and temporal features, our synthesis module is ready to successfully perform these inputs without major problems.

4. Reflection on the usability of InterL

The breadth of the SignON project is also reflected in the work on the InterL. With the development of the InterL-E we focused on one specific model, based on mBART, fine-tuned to fit multiple purposes. The results from spoken-to-spoken (text or audio) translation are within the currently-accepted norms; however, the results from the different SLMT pipeline experiments indicate that, on the one hand, the quality of such an approach is below desirable and on the other hand, that this approach, using an intermediate InterL which is given as input SLR embeddings is not surpassed by others, even end-to-end approaches. Furthermore, this approach is in line with state-of-the-art work, i.e. the submissions to the 2nd WMT shared task on SLMT. This indicates that regardless of the language pairs, there is a tremendous gap between sign language recognition and translation which, as indicated by the initial experiments we conducted on the narrow-domain Phoenix data set, is limited by the lack of data. The data requirements are also evident from the empirical evaluation of the SLR models as indicated, among others, in deliverable *D3.2 Sign language recognition component and models*.

The rule-based approaches and symbolic representations (SignNets and AMR) showed promising results (see D3.6, D4.1 and D4.2) but are limited to specific languages or bounded by the required human efforts. With the work on combining the mBART model with AMR for the purpose of SL synthesis, we

demonstrated a data-driven approach to symbolic representation which, despite the needed human efforts, is a fundamental part of our sign synthesis pipeline. Here we need to also note the work on Sign_A, which is detailed in *D5.4 First Sign language-specific lexicon and structure*, *D5.5 Second Sign language-specific lexicon and structure* and *D5.6: Final Sign language-specific lexicon and structure*. Sign_A attempts to capture signed utterances (from both syntactic and semantic perspectives) with a high level of complexity. While planned at the beginning of the project, the development of Sign_A was limited to certain utterances and used as a proof of concept as a bridge between InterL and the sign synthesis component.

5. Conclusions and future work

This deliverable presents the work on the different interlingual representations and their utilisation in translation pipelines. It also covers our empirical evaluation of those pipelines. We show that, while using our approach for translating signed to spoken languages and vice-versa via a common representation (InterL) is not effective for real world applications at this stage, it is on a par with state-of-the-art models and systems that have been fined-tuned for specific language pairs and use-cases. This leads to a two-sided conclusion – either a new, completely different approach is needed, or a lot more data is needed for the proposed method to be effective. In the future we plan to continue this work along these two dimensions: model development and data collection.

References

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., & Schneider, N. (2013). Abstract Meaning Representation for Sembanking. Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, 178–186. https://aclanthology.org/W13-2322

Camgöz N.C., Koller O, Hadfield S, Bowden R (2020) Sign language transformers: Joint end-to-end sign language recognition and translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10023–10033

Camgöz, N. C., et al. "Content4all open research sign language translation datasets." *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*. IEEE, 2021.

Crasborn, O.A.; Efthimiou, E.; Hanke, T. (ed.), Proceedings of the 3rd Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora, pp. 44-49

Docío-Fernández, Laura, et al. (2020) "Lse_uvigo: A multi-source database for Spanish sign language recognition." *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*. 2020.

De Sisto, M., Vandeghinste, V., and Shterionov, D.. 2023. A New English-Dutch-NGT Corpus for the Hospitality Domain. In Proceedings of the Second International Workshop on Automatic Translation for Signed and Spoken Languages, pages 34–37, Tampere, Finland. European Association for Machine Translation.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019, May). Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning* (pp. 2790-2799). PMLR.

Leeson, L. and Nolan, B., (2008, June) Digital Deployment of the Signs of Ireland Corpus in Elearning, LREC 2008 Conference Proceedings, LREC, Marrakesh, Morocco, In Proceedings of LREC

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

Müller, M., Alikhani, M., Avramidis, E., Bowden, R., Braffort, A., Camgöz, N. C., Ebling, S., España-Bonet, C., Göhring, A., Grundkiewicz, R., Inan, M., Jiang, Z., Koller, O., Moryossef, A., Rios, A., Shterionov, D., Sidler-Miserez, S., Tissi, K., Van Landuyt, D. (2023, December). Findings of the Second WMT Shared Task on Sign Language Translation (WMT-SLT23). In *Proceedings of the Eighth Conference on Machine Translation* (pp. 68-94).

Popović, M. (2015, September). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation* (pp. 392-395).

Schembri, A. and Fenlon, J. and Rentelis R. and Stamp, R. and Cormier, K. (2011). British Sign Language Corpus Project: A corpus of digital video data and annotations of British Sign Language 2008-2017 (Third Edition). London: University College London.

Zettlemoyer. (2020) "Multilingual denoising pre-training for neural machine translation." *Transactions of the Association for Computational Linguistics* 8 (2020): 726-742.

Annex A: Sign-to-text experiments

In this Annex, all of the experiments carried out to translate sign language utterances into text utterances will be presented.

A.1: VGT-Dutch

All of these experiments were performed using the VGT SLR model (of hidden size 128) integrated in the application. Hence, the inputs for all of them will be SLR embeddings. These can be of type (1) spatial or (2) temporal, depending on the intermediate layer from which they are obtained in the SLR module. Besides, we can organise the inputs into various types:

1. Concatenating the embeddings of size 128 to get a matrix of size Nx128, where N is the number of embeddings or sequence length. When this approach is used, we also need to include a linear layer before mBART to transform the hidden size from 128 to 1024 (the hidden dimension used by mBART).
2. Stacking clips until size 1024 is obtained, i.e. 8 embeddings are concatenated in a single embedding to get the required size. No linear layer is required. The input matrix has a size of Nx1024.
3. Stacking the spatial and temporal embeddings so that, for each frame, its spatial (hidden size 128) and temporal (also hidden size 128) SLR representations are stacked to obtain an embedding of size 256. The input matrix has a size of Nx256 and a linear layer is required.
4. The same as the previous but L2 normalising each embedding before concatenating them.
5. Concatenating the 128 embeddings into a single sequence of size Nx128 but zero padding each embedding to get size 1024, hence the input will become of size Nx1024.

Moreover, apart from the linear layer to upscale the hidden dimension, it is also possible to include an activation function SoftSign. We will refer to this as “(SS)”. We can also include some special tokens from mBART in the input sequence, namely the End-Of-Sentence (EOS) token and the Source Language (SRC) token. These are 1024 tokens extracted from the mBART embedding table and are included in the 1024-size sequence at the end.

For some experiments we machine translated the original output (spoken) sentences to Spanish and English in order to augment the data. That is, at training time, for each input sample, we generate 3

different inputs: one translates the original SL utterance to English, another one to Spanish and the remaining one to Dutch. The evaluation is performed for Dutch only.

In the following table, we will refer to each of the inputs by the number they have associated (from 1 to 5).

Model+input	*Hyperparams*	*BLEU-1*	*BLEU-2*	*BLEU-3*	*BLEU-4*	*CHRF*	*ROUGE*
Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*0.53*	*0.17*	*0.08*	*0.00*	*9.03*	*1.19*
Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*6.36*	*1.21*	*0*	*0*	*16.321*	*4.98*
Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-5} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*5.32*	*1.55*	*0.56*	*0.26*	*13.42*	*5.05*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*

Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*6.12*	*1..52*	*0.55*	*0.26*	*15.54*	*4.91*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-5} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001*	*6.38*	*1.67*	*0.51*	*0.25*	*13.15*	*5.79*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 EOS_token=Yes SRCLANG_token=Yes*	*5.75*	*1.19*	*0.4*	*0*	*16.32*	*4.94*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-5} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 EOS_token=Yes SRCLANG_token=Yes*	*0.00*	*0.00*	*0.00*	*0.00*	*0.97*	*0.00*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=Yes L.earning rate=1e^{-4}*	*5.32*	*1.24*	*0.53*	*0.33*	*13.96*	*5.17*

	Stop metric=BLEU-4 *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *EOS_token=Yes* *SRCLANG_token=Yes*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *EOS_token=Yes* *SRCLANG_token=Yes*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
Model=Linear layer (SS)+Transformer Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Hidden_size=128* *No. heads=8* *FF_size=512*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *EOS_token=Yes*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1*	*6.59*	*1.34*	*0.00*	*0.00*	*16.21*	*5.01*

	TransBeamSizeTest=6 *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *EOS_token=Yes*						
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *EOS_token=Yes*	*5.04*	*0.96*	*0.28*	*0.00*	*13.31*	*4.74*
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *SRC_token=Yes*	*1.23*	*0.27*	*0.00*	*0.00*	*7.57*	*2.51*
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *SRC_token=Yes*	*6.35*	*1.36*	*0.35*	*0.00*	*15.81*	*5.13*
Model=mBART Input=(2)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *SRC_token=Yes*	*0.00*	*0.00*	*0.00*	*0.00*	*0.59*	*0.00*

Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*1.46*	*0.46*	*0.00*	*0.00*	*5.21*	*2.21*
Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*10.27*	*2.41*	*0.00*	*0.00*	*15.44*	*9.08*
Model=Linear layer (SS)+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-5} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*9.59*	*2.04*	*0.00*	*0.00*	*14.63*	*8.29*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=1 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*2.35*	*0.52*	*0.13*	*0.00*	*6.3*	*3.38*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=5e^{-4}*	*5.51*	*1.64*	*0.48*	*0.00*	*14.43*	*6.18*

	Stop metric=BLEU-4 *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*4.65*	*1.6*	*0.53*	*0.21*	*12.21*	*5.54*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=5e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*6.58*	*1.66*	*0.4*	*0.00*	*11.72*	*6.18*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*8.16*	*2.11*	*0.59*	*0.27*	*16.25*	*7.07*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-6}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1*	*4.73*	*1.46*	*0.59*	*0.00*	*10.84*	*6.01*

	Dropout=0.1 *Weight decay=0.001* *Output_Langs=ES,EN,NL*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=TranslationLoss* *Patience=8* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*7.56*	*1.82*	*0.54*	*0.00*	*16.02*	*6.88*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*8.00*	*2.35*	*0.73*	*0.32*	*16.14*	*7.35*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=3* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*7.77*	*2.17*	*0.61*	*0.00*	*15.84*	*7.1*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*6.67*	*1.54*	*0.66*	*0.3*	*16.00*	*5.48*
Model=Linear layer+mBART	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128*	*7.52*	*2.53*	*0.85*	*0.00*	*15.32*	*7.85*

Input=(1)	*Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*0.25*	*0.07*	*0.00*	*0.00*	*3.21*	*1.23*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*6.65*	*1.67*	*0.4*	*0.00*	*16.05*	*5.48*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*8.43*	*2.48*	*0.84*	*0.43*	*16.09*	*8.17*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2*	*10.43*	*3.01*	*1.15*	*0.44*	*16.5*	*8.94*

	TransBeamSizeTest=6 *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*						
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*4.34*	*1.03*	*0.00*	*0.00*	*17.06*	*4.79*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*7.73*	*2.05*	*0.73*	*0.35*	*18.24*	*7.66*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*8.65*	*2.79*	*0.98*	*0.4*	*16.15*	*7.88*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*2.78*	*0.88*	*0.00*	*0.00*	*9.03*	*3.62*

Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*5.07*	*1.75*	*0.57*	*0.22*	*12.93*	*6.16*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-5} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*8.47*	*2.56*	*0.68*	*0.00*	*16.21*	*7.71*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=5e^{-6} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*6.93*	*1.79*	*0.53*	*0.25*	*12.04*	*6.45*
Model=Linear layer+mBART Input=(4)	*Feature=Spatial+Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Dropout=0.1 Weight decay=0.001 Output_Langs=ES,EN,NL*	*4.43*	*1.01*	*0.00*	*0.00*	*16.95*	*4.1*
Model=Linear layer+mBART Input=(4)	*Feature=Spatial+Temporal SeqSubsampled=Yes Batch size=128 Adapters=No L.earning rate=1e^{-4}*	*7.17*	*1.86*	*0.63*	*0.00*	*15.02*	*6.54*

	Stop metric=BLEU-4 *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*						
Model=Linear layer+mBART Input=(4)	*Feature=Spatial+Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001* *Output_Langs=ES,EN,NL*	*6.94*	*2.11*	*0.86*	*0.00*	*13.11*	*7.54*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*0.67*	*0.24*	*0.00*	*0.00*	*4.12*	*4.13*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*6.39*	*1.46*	*0.46*	*0.23*	*16.04*	*5.36*
Model=Linear layer+mBART Input=(3)	*Feature=Spatial+Temporal* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*4.14*	*0.7*	*0.00*	*0.00*	*9.69*	*5.13*

Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*3.37*	*1.06*	*0.3*	*0.00*	*6.5*	*4.18*
Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*7.96*	*2.03*	*0.00*	*0.00*	*18.27*	*7.61*
Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=Yes* *Batch size=128* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*6.94*	*1.78*	*0.00*	*0.00*	*12.63*	*6.21*
Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*5.89*	*1.61*	*0.00*	*0.00*	*13.69*	*7.24*
Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1*	*5.92*	*1.54*	*0.00*	*0.00*	*15.91*	*6.15*

	Dropout=0.1 *Weight decay=0.001*						
Model=mBART Input=(5)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Dropout=0.1* *Weight decay=0.001*	*4.95*	*1.09*	*0.00*	*0.00*	*11.00*	*5.57*

A.2: NGT-Dutch

These experiments were performed using

1. The frozen embeddings from the NGT model from the SLR module integrated in the application as input to the models presented next and only training/fine-tuning the SLT model.
2. Or combining the NGT SLR module (without the gloss classification head) and the SLT models that are presented next and jointly training/fine-tuning both for SLT.

The SLR module may output embeddings of two different sizes depending on the SLR model used as backbone. The sizes are 128 and 192. The input types are explained in Section A.1.

Model+input	***Hyperparams***	***BLEU-1***	***BLEU-2***	***BLEU-3***	***BLEU-4***	***CHRF***	***ROUGE***
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=32* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*6.09*	*0.93*	*0.00*	*0.00*	*13.32*	*7.65*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=32* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20*	*6.23*	*1.32*	*0.00*	*0.00*	*12.7*	*5.67*

	TransBeamSizeDev=1 *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*						
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*7.4*	*2.05*	*0.00*	*0.00*	*13.89*	*7.84*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=5e^{-5}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*6.53*	*1.91*	*0.00*	*0.00*	*11.72*	*8.24*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=5e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*7.59*	*2.07*	*0.00*	*0.00*	*13.25*	*7.37*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=1*	*7.4*	*2.05*	*0.00*	*0.00*	*13.89*	*7.84*

	TransBeamSizeTest=6 *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001* *Dropout_after_upsample=0.1*						
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=Translation Loss* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001* *Dropout_after_upsample=0.1*	*5.21*	*0.86*	*0.00*	*0.00*	*10.24*	*5.85*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=Translation Loss* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.00001* *Dropout_after_upsample=0.1*	*6.76*	*0.97*	*0.00*	*0.00*	*10.38*	*6.26*
Model=SLR+Linear layer+mBART Input=(1)	*Feature=Temporal* *SLR_hidden_size=128* *SeqSubsampled=Yes* *Batch size=128* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=Translation Loss* *Patience=20* *TransBeamSizeDev=1* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0* *Dropout_after_upsample=0.1*	*6.74*	*0.00*	*0.00*	*0.00*	*10.93*	*8.33*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8*	*6.58*	*2.25*	*0.95*	*0.40*	*12.52*	*7.21*

	TransBeamSizeDev=2 *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*7.77*	*2.06*	*0.79*	*0.33*	*13.11*	*6.57*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*3.76*	*1.01*	*0.46*	*0.27*	*7.56*	*5.95*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*6.26*	*1.41*	*0.50*	*0.24*	*10.54*	*5.70*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*6.42*	*2.39*	*1.17*	*0.60*	*12.60*	*7.49*

Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=Yes L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Label smoothing=0 Dropout=0.1 Weight decay=0.001*	*6.76*	*2.31*	*1.07*	*0.53*	*12.99*	*7.47*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=Yes L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=3 Label smoothing=0 Dropout=0.1 Weight decay=0.001*	*6.57*	*2.46*	*1.20*	*0.61*	*12.73*	*7.59*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=Yes L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=20 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=3 Label smoothing=0 Dropout=0.1 Weight decay=0.001*	*6.44*	*2.21*	*0.95*	*0.48*	*12.30*	*7.47*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial SeqSubsampled=No Batch size=64 Adapters=Yes L.earning rate=1e^{-3} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=2 Label smoothing=0 Dropout=0.1 Weight decay=0.001*	*5.99*	*1.78*	*0.74*	*0.28*	*11.68*	*7.14*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial SeqSubsampled=No Batch size=64 Adapters=Yes L.earning rate=1e^{-4}*	*6.96*	*2.20*	*0.90*	*0.41*	*12.85*	*7.60*

	Stop metric=BLEU-4 *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=2* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*						
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=2* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*0.02*	*0.00*	*0.00*	*0.00*	*1.63*	*2.07*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-3}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=2* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*5.32*	*1.77*	*0.80*	*0.36*	*11.94*	*7.22*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=2* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*6.10*	*1.94*	*0.88*	*0.44*	*12.27*	*7.36*
Model=Linear layer+mBART Input=(1)	*Feature=Spatial* *SeqSubsampled=No* *Batch size=64* *Adapters=Yes* *L.earning rate=1e^{-5}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=2*	*6.01*	*1.88*	*0.78*	*0.35*	*10.55*	*7.19*

	Label smoothing=0 *Dropout=0.1* *Weight decay=0.001*						
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=CHRF* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*7.15*	*1.63*	*0.57*	*0.25*	*16.25*	*7.07*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=translation_loss* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*4.05*	*0.97*	*0.33*	*0.00*	*9.43*	*5.46*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=perplexity* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*4.05*	*0.97*	*0.33*	*0.00*	*9.43*	*5.46*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=64* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0.1* *Dropout=0.1* *Weight decay=0.001*	*6.49*	*1.74*	*0.62*	*0.21*	*12.58*	*7.11*

Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Label smoothing=0 Dropout=0.2 Weight decay=0.001*	*5.30*	*1.49*	*0.60*	*0.24*	*12.64*	*5.80*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=No L.earning rate=1e^{-4} Stop metric=BLEU-4 Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Label smoothing=0 Dropout=0.1 Weight decay=0.01*	*1.99*	*0.61*	*0.24*	*0.12*	*7.34*	*4.20*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=No L.earning rate=1e^{-4} Stop metric=translation_loss Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Label smoothing=0.1 Dropout=0.1 Weight decay=0.001*	*0.24*	*0.04*	*0.00*	*0.00*	*2.55*	*1.96*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=No L.earning rate=1e^{-4} Stop metric=translation_loss Patience=8 TransBeamSizeDev=2 TransBeamSizeTest=6 TransBeamAlpha=1 Label smoothing=0 Dropout=0.2 Weight decay=0.001*	*4.05*	*0.97*	*0.33*	*0.00*	*9.43*	*5.46*
Model=Linear layer+mBART Input=(1)	*Feature=Temporal SeqSubsampled=Yes Batch size=64 Adapters=No L.earning rate=1e^{-4}*	*1.92*	*0.47*	*0.20*	*0.11*	*6.28*	*4.05*

	Stop metric=translation_loss *Patience=8* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.01*						
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=100* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0*	*6.36*	*2.17*	*1.05*	*0.55*	*11.35*	*7.25*
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=100* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.001*	*5.35*	*1.50*	*0.61*	*0.31*	*10.24*	*6.45*
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=100* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0.1* *Weight decay=0.0001*	*6.10*	*1.93*	*0.94*	*0.55*	*10.76*	*7.53*
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=100* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1*	*5.75*	*1.79*	*0.83*	*0.45*	*10.52*	*7.25*

	Label smoothing=0 *Dropout=0.1* *Weight decay=0.00001*						
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=100* *Adapters=No* *L.earning rate=1e^{-4}* *Stop metric=BLEU-4* *Patience=20* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0* *Weight decay=0*	*6.22*	*1.82*	*0.80*	*0.41*	*10.83*	*7.13*
Model=Linear layer+Transforme r Input=(1)	*Feature=Temporal* *SeqSubsampled=Yes* *Batch size=32* *Adapters=Yes* *L.earning rate=5e^{-5}* *Stop metric=BLEU-4* *Patience=5* *TransBeamSizeDev=2* *TransBeamSizeTest=6* *TransBeamAlpha=1* *Label smoothing=0* *Dropout=0* *Weight decay=0*	*4.89*	*1.44*	*0.62*	*0.33*	*9.54*	*6.76*