ÚFAL CorPipe at CRAC 2023: Larger Context Improves Multilingual Coreference Resolution

Milan Straka
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské nám. 25, Prague, Czech Republic
straka@ufal.mff.cuni.cz

Abstract

We present CorPipe, the winning entry to the CRAC 2023 Shared Task on Multilingual Coreference Resolution. Our system is an improved version of our earlier multilingual coreference pipeline, and it surpasses other participants by a large margin of 4.5 percent points. CorPipe first performs mention detection, followed by coreference linking via an antecedent-maximization approach on the retrieved spans. Both tasks are trained jointly on all available corpora using a shared pretrained language model. Our main improvements comprise inputs larger than 512 subwords and changing the mention decoding to support ensembling. The source code is available at https://github.com/ufal/crac2023-corpipe.

1 Introduction

The goal of coreference resolution is to identify and cluster multiple occurrences of entities in the input text. The CRAC 2023 Shared Task on Multilingual Coreference Resolution (Žabokrtský et al., 2023) aims to stimulate research in this area by featuring coreference resolution on 17 corpora in 12 languages from the CorefUD 1.1 dataset (Novák et al., 2022). The current shared task is a reiteration of the previous year's CRAC 2022 Shared Task (Žabokrtský et al., 2022).

CorPipe, our entry to the CRAC 2023 Shared Task, is an improved version of our earlier multilingual coreference pipeline (Straka and Straková, 2022), which was the winner of the last year's shared task. Our system first performs mention detection, followed by the coreference linking via an antecedent-maximization approach on the retrieved spans. However, CorPipe is not a pure pipeline, because we train both tasks jointly using a shared pretrained language model. Performing mention detection first avoids the challenge of end-to-end systems that need to consider an overwhelming number of possible spans, and also permits recognition of single-mention entities. Finally, all our models are multilingual and are trained on all available corpora.

Our contributions are as follows:

- We present a winning entry to the CRAC 2023 Shared Task with state-of-the-art results, surpassing other shared task participants by a large margin of 4.5 percent points.
- We improve our last year's system by (a) increasing the size of the inputs during prediction, while keeping it smaller during training, (b) using larger pretrained language models, (c) proposing a different mention decoding approach, that allows (d) implementing ensembling to further improve the performance.
- We perform a thorough examination of the newly introduced components.
- The source code of our system is available at https://github.com/ufal/crac2023-corpipe.

2 Related Work

While coreference resolution was traditionally carried out by first performing mention detection followed by coreference linking (clustering), recent approaches are often end-to-end (Lee et al., 2017, 2018). Likewise, the baseline of CRAC 2022 and 2023 Shared Tasks (Pražák et al., 2021) as well as the CRAC 2022 second-best solution (Pražák and Konopik, 2022) follow this approach.

The recent work of Bohnet et al. (2023) pushes the end-to-end approach even further, solving both mention detection and coreference linking jointly via a text-to-text paradigm, reaching state-of-the-art results on the CoNLL 2012 dataset (Pradhan et al., 2012). Given that our system uses the same pretrained encoder but a custom decoder designed specifically for coreference resolution instead of a general but pretrained decoder, it would be interesting to perform a direct comparison of these systems.

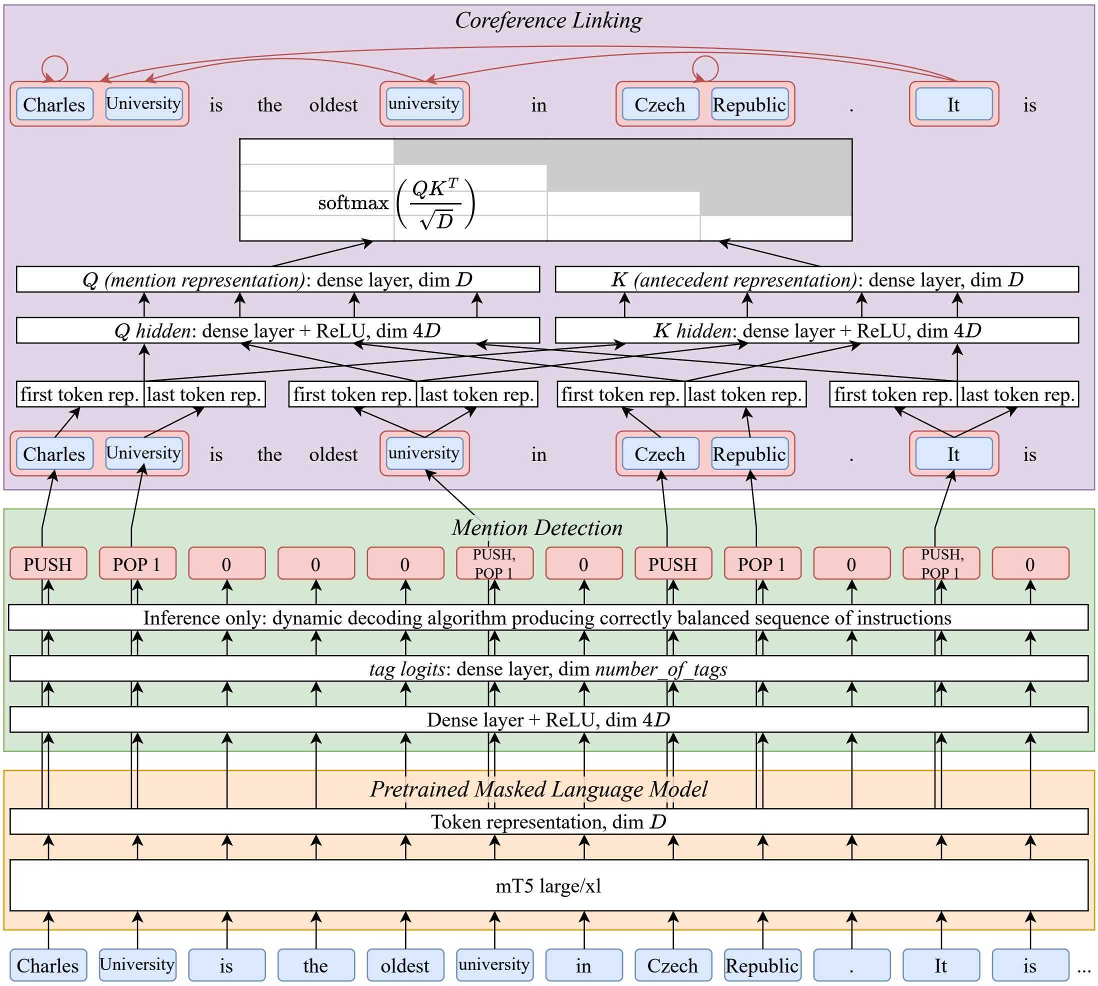

Figure 1: The proposed CorPipe model architecture.

3 CorPipe Architecture

The CorPipe architecture is based heavily on our earlier system (Straka and Straková, 2022), which won the CRAC 2022 Shared Task (Žabokrtský et al., 2022). We describe just the changes we propose; please refer to (Straka and Straková, 2022) for the description of our original system.

In short, our system first obtains a contextualized representation of the input by employing a pretrained model. These representations are then used first to perform mention detection, and then, together with the predicted mentions, to perform coreference linking. The mentions are predicted one sentence at a time, but both previous and following contexts are included up to the specified *context length*. The architecture overview is displayed in Figure 1.

3.1 The mT5 Pretrained Models

In the original architecture, we employed large-sized models XLM-R large (Conneau et al., 2020) and RemBERT (Chung et al., 2021). However, even bigger models consistently deliver better performance in various applications (Kale and Rastogi, 2020; Xue et al., 2021; Rothe et al., 2021; Bohnet et al., 2023). We therefore decided to utilize the largest possible pretrained multilingual model. To our best knowledge, we are aware of a single family of such models, the mT5 (Xue et al., 2021), a multilingual variant of the encoder-decoder pretrained model T5 (Kale and Rastogi, 2020) based on the Transformer architecture (Vaswani et al., 2017).[1]

The mT5 pretrained models have one more considerable advantage – because of relative positional embeddings, they are capable of processing inputs longer than 512 subwords, compared to both XLM-R large and RemBERT. In Section 5.1, we demonstrate that processing longer inputs is advantageous for coreference resolution.

[1] The ByT5 (Xue et al., 2022), a byte version of multilingual T5, is also available, but because it represents words as individual UTF-8 bytes, it processes smaller inputs compared to mT5, which is undesirable for coreference resolution.

3.2 Mention Decoding

In the original architecture, we reduce the representation of embedded and possibly crossing mentions to a sequence classification problem using an extension of BIO encoding. Each input token is assigned a single tag, which is a concatenation of a sequence of stack-manipulating instructions:

- any number of `POP`(i) instructions, each closing an opened mention from the stack. To support crossing mentions, any mention on the stack (not just the top one) can be closed, identified by its index i from the top of the stack (i.e., `POP`(1) closes the mention on the top of the stack, `POP`(2) closes the mention below the top of the stack);
- any number of `PUSH` instructions, each starting a new mention added to the top of the stack;
- any number of `POP`(1) instructions, each closing a single-token mention started by a `PUSH` instruction from the same tag (such single-token mentions could be also represented by a dedicated instruction like `UNIT`, but we prefer smaller number of instructions).

To produce hopefully valid (well-balanced) sequences of tags, we originally used a linear-chain conditional random fields (CRF; Lafferty et al. 2001). Because of the Markovian property, every tag had to be parametrized also with the size of the stack before the first instruction (we call these tags the *depth-dependent tags*).

The described approach has two drawbacks. First, the predicted sequence of tags might still be unbalanced (which we observed repeatedly in the predictions). Furthermore, it would be more challenging to perform ensembling, because every model would have a different sequence-based partition function.[2]

To alleviate both mentioned issues, we propose to replace the CRF with per-token classification during training and perform a constrained dynamic programming decoding during inference using the Viterbi algorithm.[3] Such approach admits ensembling in a straightforward manner by averaging predicted distributions for each token independently.

Without the CRF, the tags no longer need to be parametrized by the current size of the stack – the depth of the stack can be tracked just during decoding (we consider stack depths of at most 10; Section 5.2 demonstrates that depth 3 is actually sufficient). Such *depth-independent tags* have the advantage of being scarcer,[4] admitting better statistical efficiency, and we utilize them in our primary submission. The comparison of both tag sets as well as the CRF and dynamic programmic decoding is performed in Section 5.2.

3.3 Multilingual Training Data

All our models are trained on all 17 CorefUD 1.1 corpora. Given that their size range from tiny (457 training sentences in `de` and `en parcorfull`) to large (almost 40k training sentences in `cs pdt` and `cs pcedt`), we try to level the individual corpora performances by sub-/over-sampling the datasets. Concretely, we sample each batch example (a sentence with its context) proportionally to *mix ratios*, the corpora-specific weights. We consider the following possibilities:

- *uniform*: we sample uniformly from all corpora, ignoring their sizes;
- *linear*: we sample proportionally to the sizes of individual corpora;
- *square root*: following (van der Goot et al., 2021), we sample proportionally to the square roots of corpora sizes;
- *logarithmic*: similar to (Straka and Straková, 2022), we sample proportionally to the corpora sizes logarithms, which are linearly rescaled so that the largest corpus is ten times more probable than the smallest corpus.

Since different corpora might require particular annotations, we also consider adding a *corpus id* subword (dataset label) to the input to indicate the dataset of origin and the required style of annotations. These *corpus ids*, evaluated already in (Straka and Straková, 2022), are just a different implementation of treebank embeddings proposed in Stymne et al. (2018).

[2]When ensembling models, we average the *distributions* the models predict; in other words, unnormalized logits must first be normalized into (log-)probabilities. While this is straightforward for simple classification, CRF models normalize over all possible label sequences. Ensembling several CRF models would therefore require that, during each step of the sequential decoding of token labels, every model computed the *(log-)probabilities* of all sequences with the label in question conditioned on the already decoded labels. Such an algorithm would have the same asymptotic complexity as the usual CRF decoding times the number of models. However, we did not implement it ourselves.

[3]The decoding algorithm differs from CRF decoding in just two aspects: (a) the logits are normalized into log-probabilities for each token separately, (b) the transition matrix only forbids invalid transitions, all valid transitions have the same weight.

[4]There are 54 and 207 unique *depth-independent* and *depth-dependent tags* in the whole training data, respectively.

System	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
ÚFAL CorPipe	**74.90** **1**	**82.59** **1**	**79.33** **1**	**79.20** **1**	**72.12** **1**	**71.09** **1**	**76.57** **1**	**69.86** **1**	**83.39** **1**	**69.82** **1**	**68.92** **1**	**69.47** **1**	**75.87** **1**	**78.74** **1**	**78.77** **1**	**79.54** **1**	**82.46** **1**	**55.63** **1**
Anonymous	70.41 2	79.51 2	75.88 2	76.39 2	64.37 3	68.24 5	72.29 2	59.02 3	80.52 2	66.13 2	64.65 3	66.25 2	70.09 2	75.32 2	73.33 2	77.58 2	80.19 2	47.22 2
Ondfa	69.19 3	76.02 3	74.82 3	74.67 3	71.86 2	69.37 3	71.56 3	61.62 2	77.18 3	60.32 4	66.38 2	65.75 4	68.52 3	72.39 4	70.91 4	76.90 3	76.50 4	41.52 4
McGill	65.43 4	71.75 4	67.67 7	70.88 4	41.58 7	70.20 2	66.72 4	47.27 4	73.78 4	65.17 3	60.74 4	65.93 3	65.77 6	73.73 3	72.43 3	76.14 4	77.28 3	45.28 3
DeepBlueAI	62.29 5	67.55 7	70.38 4	69.93 5	48.81 5	63.90 7	63.58 6	43.33 5	69.52 5	55.69 6	54.38 5	63.14 5	66.75 4	69.86 6	68.53 5	73.11 5	74.41 5	36.14 8
DFKI-Adapt	61.86 6	68.21 6	68.72 5	67.34 6	52.52 4	69.28 4	65.11 5	36.87 7	69.19 6	58.96 5	51.53 7	58.56 6	66.01 5	70.05 5	68.21 6	67.98 6	72.48 6	40.67 5
Morfbase	59.53 7	68.23 5	64.89 8	64.74 8	39.96 9	64.87 6	62.80 8	40.81 6	69.01 7	53.18 8	52.91 6	56.41 7	64.08 7	68.17 7	66.35 7	67.88 7	68.53 8	39.22 6
BASELINE†	56.96 8	65.26 8	67.72 6	65.22 7	44.11 6	57.13 9	63.08 7	35.19 8	66.93 8	55.31 7	40.71 9	55.32 8	63.57 8	65.10 9	65.78 8	66.08 8	69.03 7	22.75 9
DFKI-MPrompt	53.76 9	55.45 9	60.39 9	56.13 9	40.34 8	59.75 8	57.83 9	34.32 9	58.31 9	52.96 9	44.53 8	48.79 9	56.52 9	65.12 8	62.99 9	61.15 9	61.96 9	37.44 7

Table 1: Official results of CRAC 2023 Shared Task on the test set (CoNLL score in %). The system † is described in Pražák et al. (2021); the rest in Žabokrtský et al. (2023).

Our primary submission relies on *logarithmic* mix ratios with *corpus ids*. The concrete values of all proposed mix ratios together with their performance comparison are presented in Section 5.5.

3.4 Training

When utilizing the mT5 pretrained models, we train CorPipe models with the Adafactor optimizer (Shazeer and Stern, 2018) using a slanted triangular learning schedule – we first linearly increase the learning rate from 0 to `5e-4` in the first 10% of the training, and then linearly decay it to 0 at the end of the training. The models are trained for 15 epochs, each comprising 8000 batches. For models up to size large, we utilize batch size 8, which is the maximum one fitting on a single A100 GPU with 40GB RAM. The xl-sized models are trained on four 40GB A100, with a maximum possible batch size 12. The training took 10 and 20 hours for the mT5-large and mT5-xl models, respectively.

For the XLM-R and RemBERT ablation experiments, we utilize the lazy variant of the Adam optimizer (Kingma and Ba, 2015) and the learning rates of `2e-5` and `1e-5`, respectively.

All classification heads employ label smoothing (Szegedy et al., 2016) of 0.2.

During training, we use *context length* of 512 subwords and limit the right context length to 50, but we use *context length* of 2560 subwords during inference with the mT5 models.

The competition submissions were selected from a pool of 30 models based on mT5-large and mT5-xl pretrained models with different random seeds and slightly perturbed hyperparameters,[5] by considering for each corpus the best performing checkpoint of every epoch of every trained model. Our primary submission is for each corpus an ensemble of 3 best checkpoints of 3 models.[6]

[5] Learning rate `5e-4`, `6e-4`, `7e-4`; double or quadruple batch size; 8k or 10k batches per epoch.

System	Head-match	Partial-match	Exact-match	+Singletons
ÚFAL CorPipe	**74.90 (1)**	**73.33 (1)**	**71.46 (1)**	**76.82 (1)**
Anonymous	70.41 (2)	69.23 (2)	67.09 (2)	73.20 (2)
Ondfa	69.19 (3)	68.93 (3)	53.01 (8)	68.37 (3)
McGill	65.43 (4)	64.56 (4)	63.13 (3)	68.23 (4)
DeepBlueAI	62.29 (5)	61.32 (5)	59.95 (4)	54.51 (5)
DFKI-Adapt	61.86 (6)	60.83 (6)	59.18 (5)	53.94 (6)
Morfbase	59.53 (7)	58.49 (7)	56.89 (6)	52.07 (7)
BASELINE	56.96 (8)	56.28 (8)	54.75 (7)	49.32 (8)
DFKI-MPrompt	53.76 (9)	51.62 (9)	50.42 (9)	46.83 (9)

Table 2: Official results of CRAC 2023 Shared Task on the test set with various metrics in %.

4 Shared Task Results

The official results of the CRAC 2023 Shared Task are presented in Table 1. Our CorPipe system delivers the best overall score of 74.9%, surpassing the other participants by a large margin of 4.5 percent points, and also achieves the best scores for all individual corpora.

4.1 Results of Additional Metrics

The CRAC 2023 Shared Task primary metric employs *head matching*, where a predicted mention is considered correct if it has the same mention head as the gold mention, and excludes *singletons*. Comparison with other metrics is performed in Table 2. Apart from the head matching, the organizers evaluated also *partial matching* (a predicted mention is correct if it is a subsequence of the gold mention

[6] We implemented ensembling by loading each model to its dedicated A100 GPU, thus parallelizing the execution of the individual models.

Submission	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
Original CorPipe 2022	70.3	79.9	76.0	76.8	63.3	**72.6**	72.3	57.6	81.2	65.4	66.2	65.4	68.6	75.4	73.6	79.0	78.4	42.5
Single mT5 large model	+2.6	+2.2	+2.1	+0.8	+6.7	–1.2	+1.6	+4.0	+0.9	+0.1	+1.6	+3.3	**+7.4**	**+3.5**	+2.2	–0.5	+2.4	+7.6
Single mT5 xl model	+2.7	+2.0	+2.0	+1.5	+2.7	–3.0	+2.9	+6.8	+1.6	+2.6	–0.7	**+4.1**	+4.7	+3.3	+3.7	–0.3	+2.6	+10.3
Per-treebank best mT5 model	+3.4	+2.6	+1.7	+1.6	**+13.1**	–4.1	+3.2	+10.3	+1.2	+3.3	–0.2	+2.0	+6.6	+3.0	+4.2	–0.8	+3.8	+7.6
Per-treebank 3-model ensemble	**+4.6**	**+2.7**	**+3.3**	**+2.4**	+8.8	–1.5	**+4.3**	**+12.3**	**+2.2**	**+4.4**	**+2.7**	+4.1	+7.3	+3.3	**+5.2**	**+0.5**	**+4.1**	**+13.1**
Per-treebank 8-model ensemble	***+4.9***	***+3.3***	***+3.3***	***+2.7***	*+7.7*	*–0.8*	*+4.2*	***+13.4***	***+2.3***	*+3.2*	***+3.3***	***+5.4***	***+7.8***	***+4.2***	***+5.4***	***+0.8***	***+4.2***	***+14.0***

Table 3: Official results of ablation experiments on the test set (CoNLL score in %). The 8-model ensemble (in italics) was evaluated during the post-competition phase.

and contains the gold mention head), *exact matching* (a predicted mention is correct if it is exactly equal to the gold mention), and head matching including *singletons* (entities with a single mention).

The ranking of all systems is unchanged in all evaluated metrics, with a single exception – the system *Ondfa* exhibits low exact-matching performance, presumably because it reduces predicted mentions to just their heads.[7]

4.2 Results of Our Additional Submissions

To quantify this year's CorPipe improvements, we present the official results of our additional submissions in Table 3.

We first trained the original CorPipe on this year's data, achieving a 70.3% CoNLL score, which is 0.1 percent points below the second-best submission. Incorporating mT5-large/mT5-xl models, context size of 2560, and constrained decoding with depth-independent tags resulted in an increase of 3.4 percent points. Furthermore, employing a 3-model ensemble provides another 1.2 percent points raise. In the post-competition phase, we also evaluated an 8-model ensemble, which delivered a final modest improvement of 0.3 percent points and reached our best performance of 75.2%.

All these submissions choose the best model checkpoints for every corpus independently. However, for deployment, a single checkpoint is more appropriate – therefore, we also assessed the single best-performing mT5-large checkpoint, resulting in a 72.9% score (0.8 percent points lower than choosing the best mT5-large/mT5-xl checkpoint per corpus). The single best-performing mT5-xl checkpoint achieved very similar performance of 73.0%. We note that these single-checkpoint submissions would comfortably win the shared task too.

[7]Reducing mentions to heads was a strategy for improving partial-matching score in the previous edition of the shared task; with the head-matching score, it can be avoided, which allows also correct evaluation of the exact matching.

5 Ablations on the Development Set

To evaluate the effect of various hyperparameters, we perform further experiments on the development set. Because we observed a significant variance with different random seeds and we also observed divergence in some training runs, we devised the following procedure to obtain credible results: For each configuration, we perform 7 training runs and keep only the 5 ones with the best overall performance. We then want to perform early stopping for every corpus. However, choosing for every corpus a different epoch in every run could lead to maximization bias in case the results oscillate considerably – therefore, for every corpus, we choose the single epoch achieving the highest average 5-run score (i.e., we use this epoch for all 5 runs). Finally, we either average or ensemble the 5 runs for every corpus.

5.1 Pretrained Models and Context Sizes

The effect of increasing context sizes on the mT5-large pretrained model is presented in Table 4.A. The performance improves consistently with increasing context size up to 2560; however, context size 4096 deteriorates the performance slightly. Considering context size 512, decreasing the context size by 128 to 384 decreases the performance by 1.6 percent points, while increasing the context size by 128 to 768 increases it by 1.2 percent points, with performance improving up to 2 percent points for context length 2560.

For the mT5-xl pretrained model, the behavior is virtually analogous, as captured by Table 4.B.

In Table 4.C, we compare the performance of different pretrained models using the context size 512. We include different sizes of the mT5 model (Xue et al., 2021), together with RemBERT (Chung et al., 2021), XLM-R base, and XLM-R large (Conneau et al., 2020).[8]

[8]We do not include other base-sized models like XLM-V (Liang et al., 2023) or mDeBERTaV3 (He et al., 2023), because they lack behind the large-sized models.

Configuration	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
A) CONTEXT SIZES FOR THE MT5-LARGE MODEL																		
mT5-large 512	72.8	78.1	78.1	76.9	**70.7**	**75.4**	75.6	**67.4**	80.3	68.6	**70.6**	67.3	77.4	77.8	78.7	75.8	71.1	48.6
mT5-large 256	−5.9	−8.8	−4.0	−5.3	−7.1	−3.2	−5.3	−11.7	−6.0	−4.1	−2.9	−4.5	−8.6	−6.4	−6.4	−4.8	−6.7	−4.6
mT5-large 384	−1.6	−2.9	−1.3	−1.8	−0.6	−0.3	−2.0	−1.6	−2.2	−1.3	−1.4	−1.1	−2.7	−2.4	−2.6	−1.2	−2.0	−1.5
mT5-large 768	+1.2	+2.5	+1.2	+1.5	−0.7	**+0.0**	+0.9	−1.4	+1.5	+1.3	−0.6	+2.1	+0.4	+2.7	+2.2	+0.4	+2.7	+3.3
mT5-large 1024	+1.6	+3.2	+1.8	+1.9	−1.0	**+0.0**	+1.1	−1.4	+2.1	+1.7	−1.1	+2.3	**+0.5**	+3.5	+2.6	+0.7	+3.6	+4.7
mT5-large 1536	+1.9	+3.3	+2.2	+2.1	−1.0	**+0.0**	+1.2	−1.4	+2.4	+1.5	−1.1	+2.4	**+0.5**	**+3.8**	**+3.1**	+1.0	+4.1	+6.8
mT5-large 2048	+2.0	+3.5	+2.2	**+2.1**	−1.0	**+0.0**	**+1.2**	−1.4	+2.5	**+2.0**	−1.1	+2.4	**+0.5**	+3.8	+3.0	+1.2	+4.1	+7.4
mT5-large 2560	**+2.0**	**+3.5**	**+2.2**	+2.1	−1.0	**+0.0**	**+1.2**	−1.4	**+2.5**	+1.7	−1.1	**+2.5**	**+0.5**	+3.7	+3.0	**+1.3**	+4.1	**+8.6**
mT5-large 4096	+1.7	+3.4	+2.1	+2.0	−1.0	**+0.0**	**+1.2**	−1.4	+2.5	+1.5	−1.1	**+2.5**	**+0.5**	+3.7	+2.8	+1.2	**+4.4**	+3.1
B) CONTEXT SIZES FOR THE MT5-XL MODEL																		
mT5-xl 512	73.3	77.5	78.4	77.2	**73.9**	76.1	75.4	72.9	80.1	68.4	**70.3**	67.2	77.2	77.7	78.3	76.1	71.3	47.6
mT5-xl 256	−6.1	−8.6	−3.9	−5.4	−9.2	−3.7	−5.8	−9.6	−5.7	−4.9	−2.8	−4.6	−10.1	−6.1	−6.5	−4.7	−6.7	−4.7
mT5-xl 384	−1.7	−2.6	−1.3	−1.9	−2.4	+0.1	−1.6	−0.4	−2.2	−1.5	−1.6	−1.2	−2.5	−2.2	−2.3	−1.3	−2.5	−0.6
mT5-xl 768	+1.1	+2.2	+1.3	+1.7	−4.4	+0.1	+1.3	+0.9	+1.7	+1.5	−1.3	+1.9	**+1.5**	+2.6	+2.2	+0.5	+2.6	+2.4
mT5-xl 1024	+1.5	+3.2	+1.9	+2.3	−4.4	**+0.1**	+1.5	**+1.0**	+2.3	+2.1	−1.5	+2.1	+1.2	+3.3	+2.9	+0.8	+3.9	+3.2
mT5-xl 1536	+1.8	+3.4	+2.4	+2.6	−4.4	**+0.1**	+1.7	**+1.0**	+2.7	**+2.1**	−1.5	+2.2	+1.2	**+3.8**	+3.5	+1.1	+5.2	+3.5
mT5-xl 2048	+1.8	**+3.5**	+2.6	**+2.6**	−4.4	+0.1	+1.7	**+1.0**	+2.8	+2.1	−1.5	**+2.2**	+1.2	+3.7	**+3.9**	+1.3	**+5.5**	+3.6
mT5-xl 2560	**+1.9**	+3.4	+2.6	+2.6	−4.4	+0.1	+1.7	**+1.0**	+2.8	+2.0	−1.5	+2.2	+1.2	+3.7	+3.6	**+1.4**	+5.3	**+5.7**
mT5-xl 4096	+1.7	+3.5	**+2.6**	+2.5	−4.4	+0.1	**+1.7**	**+1.0**	**+2.8**	+1.8	−1.5	+2.2	+1.2	+3.6	+3.6	+1.4	+5.3	+2.6
C) PRETRAINED LANGUAGE MODELS WITH CONTEXT SIZE 512																		
mT5-large 512	72.8	78.1	78.1	76.9	70.7	75.4	75.6	67.4	80.3	68.6	**70.6**	67.3	77.4	77.8	78.7	75.8	71.1	**48.6**
mT5-small 512	−9.7	−10.2	−11.3	−11.9	−10.6	−11.9	−8.0	−2.8	−9.5	−8.4	−12.7	−8.6	−8.1	−7.0	−9.2	−11.2	−11.6	−12.8
mT5-base 512	−3.9	−4.2	−4.1	−4.5	−3.8	−5.2	−3.8	+1.2	−3.6	−3.3	−8.3	−3.8	−1.6	−3.3	−3.0	−4.3	−4.6	−7.1
XLM-R-base 512	−1.9	−2.8	−3.4	−4.0	−0.5	−3.9	−3.5	+2.4	−2.6	−1.5	−2.8	−1.7	+0.9	−1.8	−2.3	−3.3	−0.8	−2.3
XLM-R-large 512	**+1.1**	**+1.2**	+0.7	**+0.9**	+1.5	+0.8	**+0.8**	+2.7	**+0.9**	**+1.7**	−0.9	**+2.7**	**+1.0**	**+1.2**	**+1.0**	+0.6	+2.1	−0.8
RemBERT 512	+0.2	+0.7	**+1.2**	+0.7	**+3.4**	**+2.5**	+0.1	+4.2	+0.5	+1.0	−3.3	+0.0	−1.1	+0.0	+0.0	**+0.9**	**+2.2**	−10.0
mT5-xl 512	+0.5	−0.6	+0.3	+0.3	+3.2	+0.7	−0.2	**+5.5**	−0.2	−0.2	−0.3	−0.1	−0.2	−0.1	−0.4	+0.3	+0.2	−1.0
D) COMPARISON OF PRETRAINED LANGUAGE MODELS WITH DIFFERENT CONTEXT SIZES																		
mT5-large 512	72.8	78.1	78.1	76.9	70.7	75.4	75.6	67.4	80.3	68.6	**70.6**	67.3	77.4	77.8	78.7	75.8	71.1	48.6
mT5-base 512	−3.9	−4.2	−4.1	−4.5	−3.8	−5.2	−3.8	+1.2	−3.6	−3.3	−8.3	−3.8	−1.6	−3.3	−3.0	−4.3	−4.6	−7.1
XLM-R-base 256	−7.3	−10.0	−6.6	−8.0	−15.1	−5.5	−7.1	−9.8	−7.6	−4.6	−4.4	−4.7	−8.0	−6.3	−8.5	−6.5	−6.9	−5.3
XLM-R-base 384	−4.0	−5.2	−5.0	−5.6	−3.2	−4.1	−5.0	−2.2	−4.9	−2.9	−5.3	−2.8	−2.6	−3.8	−5.2	−3.8	−3.9	−2.5
XLM-R-base 512	−1.9	−2.8	−3.4	−4.0	−0.5	−3.9	−3.5	+2.4	−2.6	−1.5	−2.8	−1.7	+0.9	−1.8	−2.3	−3.3	−0.8	−2.3
XLM-R-base mT5-512	*−3.4*	*−4.9*	*−5.0*	*−5.6*	*−3.4*	*−4.1*	*−4.4*	*−0.6*	*−4.6*	*−2.3*	*−5.0*	*−3.5*	*+0.1*	*−2.9*	*−3.9*	*−3.6*	*−2.3*	*−2.2*
XLM-R-large 256	−3.9	−6.0	−2.8	−3.5	−7.6	−2.1	−3.9	−2.3	−4.1	−2.6	−2.3	−0.7	−7.6	−3.8	−5.0	−2.4	−4.6	−5.3
XLM-R-large 384	−0.7	−1.0	−0.6	−0.5	−1.6	+0.2	+0.0	+1.6	−1.3	+0.1	−2.1	+1.5	−2.5	−1.2	−1.8	+0.0	−0.9	−3.4
XLM-R-large 512	+1.1	+1.2	+0.7	+0.9	+1.5	+0.8	+0.8	+2.7	+0.9	+1.7	−0.9	**+2.7**	**+1.0**	+1.2	+1.0	+0.6	+2.1	−0.8
XLM-R-large mT5-512	*−0.1*	*−0.9*	*−0.6*	*−0.6*	*+0.5*	*+0.4*	*+0.0*	*+2.3*	*−0.9*	*+0.8*	*−2.1*	*+0.8*	*−0.7*	*+0.2*	*−0.4*	*+0.3*	*+0.5*	*−3.0*
RemBERT 256	−4.9	−7.3	−2.4	−3.9	−4.2	+1.0	−4.5	−4.7	−5.4	−3.0	−5.9	−3.5	−9.9	−5.8	−6.3	−3.1	−4.1	−11.3
RemBERT 384	−1.5	−1.9	−0.1	−0.8	+1.1	**+2.8**	−1.5	+0.8	−1.9	−0.3	−5.3	−1.1	−3.6	−2.6	−2.0	−0.1	−0.4	−9.5
RemBERT 512	+0.2	+0.7	+1.2	+0.7	+3.4	+2.5	+0.1	+4.2	+0.5	+1.0	−3.3	+0.0	−1.1	+0.0	+0.0	+0.9	+2.2	−10.0
RemBERT mT5-512	*−0.6*	*−1.0*	*+0.1*	*−0.6*	***+5.4***	*+2.6*	*−0.5*	*+2.3*	*−1.3*	*+0.4*	*−5.4*	*−0.3*	*−1.2*	*−1.0*	*−0.5*	*+0.7*	*+0.5*	*−10.5*
mT5-large 768	+1.2	+2.5	+1.2	+1.5	−0.7	+0.0	+0.9	−1.4	+1.5	+1.3	−0.6	+2.1	+0.4	+2.7	+2.2	+0.4	+2.7	+3.3
mT5-large 2560	+2.0	**+3.5**	+2.2	+2.1	−1.0	+0.0	+1.2	−1.4	+2.5	+1.7	−1.1	+2.5	+0.5	**+3.7**	+3.0	+1.3	+4.1	**+8.6**
mT5-xl 512	+0.5	−0.6	+0.3	+0.3	+3.2	+0.7	−0.2	+5.5	−0.2	−0.2	−0.3	−0.1	−0.2	−0.1	−0.4	+0.3	+0.2	−1.0
mT5-xl 2560	**+2.4**	+2.8	**+2.9**	**+2.9**	−1.2	+0.8	**+1.5**	**+6.5**	**+2.6**	**+1.8**	−1.8	+2.1	+1.0	+3.6	**+3.2**	**+1.7**	**+5.5**	+4.7

Table 4: Ablation experiments evaluated on the development sets (CoNLL score in %). We report the average of best 5 out of 7 runs, using for every corpus the single epoch achieving the highest average 5-run score. The runs in italics use largest context length not exceeding 512 subwords when tokenized with the mT5 tokenizer.

As expected, the increasingly bigger mT5 models improve the performance. Somewhat surprisingly, the XLM-R-base surpasses mT5-base and XLM-R-large and RemBERT surpass mT5-large. However, we discovered that the difference is caused primarily by different tokenization: The mT5 tokenizer produces on average more subwords than the XLM-R and RemBERT tokenizers, which effectively decreases the context size of the mT5 models – but the performance is considerably dependent on the context size.

To expose the issue, Table 4.D compares various pretrained models with different context sizes. Most importantly, we include the performance of the XLM-R and RemBERT models using a context that would be tokenized into 512 subwords by the mT5 tokenizer (presented in italics and denoted by the *mT5-512* context size). In these cases, the performance is quite similar to the performance of the corresponding mT5 model (with the notable exception of RemBERT's performance on Turkish, which is considerably worse). However, the mT5 models support larger context sizes (due to relative positional embeddings); already with context size 768, the mT5 models surpass all models of corresponding size and context size 512, ultimately providing the best results.

Configuration	Label Smoothing	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
A) CONSTRAING DECODING WITH VARYING DEPTH AND DEPTH-INDEPENDENT TAGS																			
Depth 10	*0.2*	**74.8**	**81.6**	80.3	**79.0**	69.7	75.4	**76.8**	66.0	82.8	**70.3**	**69.5**	69.8	77.9	81.5	81.7	77.1	75.2	57.2
Depth 3	*0.2*	+0.0	−0.1	**+0.0**	−0.1	+0.0	+0.0	+0.0	+0.0	−0.3	**+0.0**	**+0.0**	+0.0	+0.0	−0.1	**+0.0**	+0.0	+0.0	**+0.0**
Depth 2	*0.2*	−0.2	−0.7	−0.6	−0.9	+0.4	**+0.5**	−0.4	+0.0	−0.9	−0.1	+0.0	+0.0	+0.0	−0.2	−0.1	−0.4	+0.0	**+0.0**
Depth 1	*0.2*	−2.3	−5.9	−5.8	−6.1	−2.3	−1.1	−3.5	−0.4	−7.0	−1.3	−0.7	**+0.1**	+0.2	−2.0	−1.1	−1.9	−0.5	−0.6
Depth 10	*0.0*	−0.1	−0.4	−0.3	−0.2	+1.3	−0.8	−0.6	+0.0	−0.2	−0.2	−1.1	+0.0	+1.0	−0.1	−0.8	−0.1	**+1.1**	−0.6
Depth 3	*0.0*	−0.1	−0.5	−0.3	−0.2	**+1.3**	−0.8	−0.6	+0.0	−0.4	−0.2	−1.1	+0.0	+1.0	−0.1	−0.8	−0.2	**+1.1**	−0.6
Depth 2	*0.0*	−0.3	−1.0	−0.8	−1.0	+1.3	−0.5	−1.0	+0.0	−1.3	−0.2	−1.0	+0.0	+1.0	−0.1	−0.8	−0.6	+1.1	−0.6
Depth 1	*0.0*	−2.5	−6.7	−5.8	−6.3	−1.7	−2.2	−4.8	+0.2	−7.9	−1.6	−1.0	+0.1	**+1.0**	−1.7	−1.5	−2.2	+0.7	−1.1
Depth 10	*0.1*	−0.2	−0.1	−0.2	−0.2	+0.2	+0.2	−0.4	+0.1	**+0.2**	−0.1	−1.4	−0.5	+0.5	**+0.1**	−0.5	**+0.1**	+0.0	−1.6
Depth 3	*0.1*	−0.2	−0.2	−0.2	−0.3	+0.2	+0.2	−0.5	+0.1	+0.0	−0.1	−1.4	−0.5	+0.5	+0.0	−0.5	+0.0	+0.0	−1.6
Depth 2	*0.1*	−0.5	−0.8	−0.7	−1.1	+0.2	+0.4	−0.9	+0.1	−0.8	−0.2	−1.4	−0.5	+0.5	+0.0	−0.7	−0.5	+0.0	−1.6
Depth 1	*0.1*	−2.5	−6.2	−5.9	−6.2	−1.8	−0.9	−4.1	**+0.5**	−7.2	−1.4	−1.7	−0.4	+0.6	−1.8	−1.6	−2.0	−0.5	−2.0
B) COMPARISON OF DIFFERENT DECODING STRATEGIES																			
Constraint decoding, depth 10, depth-independent tags	*0.2*	**74.8**	81.6	80.3	79.0	69.7	75.4	76.8	66.0	**82.8**	70.3	69.5	69.8	77.9	81.5	81.7	77.1	75.2	**57.2**
Greedy, depth-dependent tags	*0.0*	−1.3	−1.1	−1.1	−1.3	−4.6	−0.3	−0.8	−1.5	−1.0	−0.7	−2.4	−1.0	−1.3	−0.8	−0.4	−0.4	−0.2	−3.1
+ constraint decoding	*0.0*	−0.4	−0.6	−0.2	+0.1	−1.6	+0.7	−0.4	−0.1	−0.4	−0.5	−0.5	−0.1	−0.6	−0.5	−0.1	−0.2	−0.3	−1.2
Greedy, depth-dependent tags	*0.1*	−1.3	−1.2	−1.2	−1.4	−3.2	−1.2	−1.0	−7.7	−1.1	−0.1	−1.6	−0.9	+0.5	−0.2	−0.1	−0.1	**+1.4**	−2.6
+ constraint decoding	*0.1*	−0.3	−0.6	−0.4	−0.1	+1.3	−0.1	−0.6	−4.9	−0.5	**+0.2**	**+0.9**	−0.1	**+0.7**	+0.1	**+0.0**	**+0.2**	+1.2	−2.2
Greedy, depth-dependent tags	*0.2*	−1.3	−1.3	−0.9	−1.2	−2.3	−1.0	−0.8	+0.8	−1.1	−0.2	−3.1	−1.1	−2.0	−1.3	−0.6	−0.7	−0.1	−5.4
+ constraint decoding	*0.2*	−0.3	−1.0	−0.3	+0.0	**+2.5**	−0.6	−0.4	**+3.3**	−0.4	+0.0	−0.9	−0.4	−0.3	−0.9	−0.3	−0.5	+0.0	−4.8
Conditional random fields	*0.0*	−0.2	−0.4	−0.3	−0.1	+1.7	−0.7	+0.0	+1.5	−0.5	−0.6	−0.3	+0.3	+0.4	−0.9	−0.4	−0.4	−0.3	−2.2
+ constraint decoding	*0.0*	−0.1	−0.3	−0.3	+0.0	+1.7	−0.6	+0.0	+1.8	−0.3	−0.6	−0.2	**+0.3**	+0.5	−1.0	−0.5	−0.4	−0.3	−2.2
Conditional random fields	*0.1*	−0.2	−0.4	**+0.1**	+0.3	+0.3	−1.1	+0.2	+1.1	−0.1	−0.3	−0.3	−0.2	−0.3	−0.2	−0.1	+0.0	+0.6	−3.6
+ constraint decoding	*0.1*	−0.2	−0.3	+0.1	**+0.4**	+0.5	−1.2	**+0.2**	+0.6	−0.1	−0.2	−0.3	−0.2	−0.2	−0.1	−0.1	−0.1	+0.5	−3.6
Conditional random fields	*0.2*	−0.3	+0.2	−0.3	+0.0	−1.2	+1.1	+0.1	+0.1	−0.2	+0.0	+0.0	+0.0	−1.5	+0.2	+0.0	+0.0	+0.9	−3.9
+ constraint decoding	*0.2*	−0.2	**+0.2**	−0.3	+0.1	−1.4	**+1.2**	+0.1	+0.4	−0.1	+0.1	+0.2	+0.0	−1.5	**+0.2**	−0.1	+0.0	+0.8	−3.9

Table 5: Ablation experiments evaluated on the development sets (CoNLL score in %) using the mT5-large model with context size 2560. We report the average of best 5 out of 7 runs, using for every corpus the single epoch achieving the highest average 5-run score.

5.2 Mention Decoding Algorithms

The effects of the mention decoding algorithm and label smoothing are elaborated in Table 5. First, label smoothing has very little effect on the results.

When predicting mentions via depth-independent tags, the maximum possible number of opened multi-word mentions (*depth*) must be specified. The effect of using depths 1, 2, 3, and 10 is presented in Table 5.A. While the maximum depth in the training data is 12, the performance of using depth 10 and 3 is virtually unchanged; only depth 2 and depth 1 deteriorate performance. If the speed of the decoding is an issue, using depth 3 provides the fastest decoder without decreasing performance.

The difference between using depth-independent and depth-dependent tags during constrained decoding is quantified in Table 5.B – depth-independent tags provide a minor improvement of 0.3 percent points. When greedy decoding is used instead of constrained decoding, the performance drops by one percent point.

Using conditional random fields for mention decoding provides marginally worse performance compared to using constrained decoding with depth-independent tags. Furthermore, explicitly disallowing invalid transitions (by assigning them transition weight $-\infty$ in the transition weight matrix manually) has virtually no effect, demonstrating that the CRF decoder has learned the transition weights successfully.

5.3 The Effect Of Multilingual Data

In Table 6, we analyze the effect of using various combinations of corpora during training.

Compared to using all corpora for single-model training, relying solely on the training data of a given corpus deteriorates the performance dramatically by 3.7 percent points on average. The decrease is smallest for the largest corpora (Czech and Polish ones).

Concatenating all corpora of a given language (and both ParCorFull corpora that are translations of each other; we utilized uniform mix ratios) generally improves the performance compared to using the individual corpora, but does not reach the performance of using all corpora together.

5.4 Zero-shot Multilingual Evaluation

When training without the corpus ids, the model is able to perform prediction on unknown languages. Leveraging this observation, we perform zero-shot

Configuration	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
Single Multilingual Model	**74.8**	**81.6**	**80.3**	**79.0**	**69.7**	**75.4**	**76.8**	**66.0**	**82.8**	**70.3**	**69.5**	**69.8**	**77.9**	**81.5**	**81.7**	**77.1**	**75.2**	**57.2**
Per-Corpus Models	–3.7	–1.4	–0.5	–0.4	–7.7	–3.3	–1.6	–7.6	–1.5	–2.0	–9.1	–1.0	–3.0	–2.3	–2.9	–1.0	–2.0	–15.8
Joint Czech Model			–0.1	–0.3														
Joint German Model					–4.8	–3.9												
Joint English Model							–1.9	–4.5										
Joint Parcorfull Model					–4.4			–2.5										
Joint Hungarian Model											–5.9	–1.1						
Joint Norwegian Model														–1.3	–1.8			
Zero-Shot Multilingual Models	–13.2	–4.8	–24.2	–16.0	–13.7	–10.6	–14.4	–13.8	–1.9	–5.4	–15.1	–15.0	–23.4	–14.3	–18.0	–17.5	–15.5	–0.8

Table 6: Ablation experiments evaluated on the development sets (CoNLL score in %) using the mT5-large model with context size 2560. We report the average of best 5 out of 7 runs, using for every corpus the single epoch achieving the highest average 5-run score.

Configuration	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
MIX RATIO WEIGHTS OF INDIVIDUAL CORPORA IN PERCENTS																		
Logarithmic		*8.1*	*10.0*	*9.4*	*1.0*	*3.2*	*6.6*	*1.0*	*8.3*	*7.4*	*2.6*	*5.8*	*3.4*	*7.2*	*6.9*	*8.6*	*6.2*	*4.2*
Uniform		*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*	*5.9*
Square Root		*8.4*	*14.0*	*11.7*	*1.4*	*2.4*	*5.6*	*1.4*	*8.8*	*6.9*	*2.0*	*4.6*	*2.5*	*6.5*	*6.0*	*9.5*	*5.1*	*3.1*
Linear		*8.7*	*24.4*	*17.0*	*0.2*	*0.7*	*3.9*	*0.2*	*9.6*	*5.9*	*0.5*	*2.6*	*0.8*	*5.3*	*4.5*	*11.3*	*3.2*	*1.2*
A) AVERAGE OF 5 RUNS USING FOR EVERY CORPUS THE SINGLE EPOCH ACHIEVING THE HIGHEST AVERAGE 5-RUN SCORE																		
Logarithmic	74.8	81.6	80.3	79.0	69.7	75.4	**76.8**	66.0	82.8	70.3	69.5	**69.7**	77.9	81.5	81.7	77.1	75.2	**57.2**
w/o corpus id	–0.2	**+0.2**	–0.1	+0.1	–0.4	+0.1	–0.3	–0.2	+0.0	+0.0	–0.2	–0.3	+0.5	**+0.2**	–0.4	**+0.2**	+0.2	–2.4
Uniform	–0.3	–0.1	–1.2	–0.9	+1.7	+0.0	–0.8	–4.2	–0.3	+0.1	**+0.2**	–0.4	**+1.0**	+0.0	–0.1	+0.0	–0.2	–0.1
w/o corpus id	–0.4	–0.4	–0.7	–0.6	+2.3	+0.3	–0.8	+1.5	–0.1	–0.4	–1.3	–0.5	–0.7	–0.4	–1.3	–0.5	–0.2	–3.0
Square Root	+0.0	+0.2	+0.5	+0.4	–0.2	**+0.9**	–0.6	–2.1	–0.1	+0.1	–0.7	–0.1	+0.8	+0.1	–0.2	+0.2	+0.9	–0.7
w/o corpus id	+0.2	+0.1	+0.4	+0.3	**+2.7**	–0.9	–0.3	+1.1	+0.1	+0.0	–0.4	–0.2	+0.1	+0.1	–0.1	+0.1	+0.5	–0.7
Linear	**+0.4**	+0.1	**+0.8**	**+0.7**	+0.6	–0.1	–0.2	**+4.8**	**+0.3**	**+0.4**	–0.9	–0.4	+0.6	–0.3	**+0.1**	+0.2	**+1.1**	–0.3
w/o corpus id	+0.0	+0.0	+0.7	+0.6	–2.0	–1.4	–0.8	+4.0	+0.3	–0.1	–0.4	–0.9	+0.4	+0.1	–0.1	+0.2	+0.7	–0.8
B) AVERAGE OF 5 RUNS USING FOR EVERY RUN THE SINGLE EPOCH ACHIEVING THE HIGHEST SCORE ACROSS ALL CORPORA																		
Logarithmic	74.8	81.7	79.9	78.6	71.5	**76.2**	**76.6**	67.9	82.8	70.4	68.3	69.4	78.0	81.4	81.5	76.9	74.6	55.5
w/o corpus id	–0.2	+0.0	+0.1	+0.2	–1.9	–0.3	–0.3	–0.9	–0.2	–0.4	+0.0	–0.2	–0.2	+0.1	–0.2	+0.3	+1.0	–0.3
Uniform	–0.6	–0.4	–1.1	–0.9	+0.1	–1.0	–0.8	–6.7	–0.4	–0.2	**+1.0**	**+0.1**	–0.2	–0.1	+0.2	–0.1	+0.5	+0.0
w/o corpus id	–0.6	–0.7	–0.6	–0.5	+1.0	–1.6	–0.5	–0.6	–0.1	–0.6	+0.3	–0.5	–0.9	–0.1	–1.3	–0.5	+0.8	–3.0
Square Root	–0.2	–0.1	+0.8	+0.7	–2.5	–0.2	–0.1	–4.2	–0.1	+0.0	+0.9	–0.4	+0.2	+0.3	+0.0	**+0.4**	+1.5	+0.4
w/o corpus id	+0.1	–0.2	+0.6	+0.6	**+1.3**	–2.1	–0.2	–0.7	+0.2	**+0.1**	+0.0	–0.4	–0.1	+0.2	+0.1	+0.1	+1.2	**+1.1**
Linear	**+0.3**	**+0.2**	**+1.1**	**+1.1**	–0.7	–1.9	–0.2	**+3.8**	**+0.5**	–0.1	–0.7	–0.1	+0.3	–0.4	**+0.3**	+0.1	**+1.6**	+0.0
w/o corpus id	+0.1	+0.0	+1.0	+1.0	–2.1	–2.5	–0.2	+1.3	+0.2	–0.1	+0.4	–0.5	**+0.5**	**+0.4**	+0.3	+0.4	+1.0	+0.8

Table 7: Ablation experiments evaluated on the development sets (CoNLL score in %) using the mT5-large model with context size 2560. We report the average of best 5 out of 7 runs.

evaluation by training multilingual models on corpora from all but one language and then evaluating the performance on the omitted-language corpora. The results are displayed on the last line of Table 6.

Overall, the results are significantly worse by 13.2 percent points. However, such performance is most likely better than the performance of the baseline system of Pražák et al. (2021), which has 17.9 less percent points on the test set than CorPipe.

Turkish demonstrates the smallest decrease in the zero-shot evaluation, even when it uses an alphabet with several unique characters. On the other hand, the small decrease in the performance of Catalan, Spanish, and French can be explained by similarities among these languages.

5.5 Mix Ratios of the Multilingual Data

Next, we compare the effect of various mix ratios during all-corpora training.

We consider *logarithmic*, *uniform*, *square root*, and *linear* mix ratios described in Section 3.3. First, their values normalized to percentages are presented in the first part of Table 7.

We then evaluate the effect of using a specific mix ratio and either utilizing or omitting the corpus ids during training in Table 7.A. In accordance with findings in Straka and Straková (2022), the corpus ids have no deterministic effect, and the mix ratios influence the system performance surprisingly little (with *uniform* being the worst, *logarithmic* and *square root* very similar and better, and *linear* the best). When considering the largest corpora (especially Czech, Polish, and Spanish), their performance improves with increasing mix ratios, presumably because of underfitting with small mix ratios; however, the effect on other corpora is mixed.

The evaluation methodology allows each corpus to use a checkpoint from a different epoch of the

Configuration	Avg	ca	cs pcedt	cs pdt	de parc	de pots	en gum	en parc	es	fr	hu korko	hu szege	lt	no bookm	no nynor	pl	ru	tr
A) Ensembles for the mT5-large Model for Various Context Sizes																		
Average of 5 runs, 512	72.8	78.1	78.1	76.9	70.7	75.4	75.6	**67.4**	80.3	68.6	70.6	67.3	77.4	77.8	78.7	75.8	71.1	48.6
Ensemble of 5 runs, 512	+1.0	+0.8	+0.8	+0.7	**+3.1**	**+1.3**	+0.5	–0.4	+0.8	+0.6	**+1.2**	+0.7	+1.6	+0.9	+0.9	+1.0	+1.5	+0.8
Average of 5 runs, 768	+1.2	+2.5	+1.2	+1.5	–0.7	+0.0	+0.9	–1.4	+1.5	+1.3	–0.6	+2.1	+0.4	+2.7	+2.2	+0.4	+2.7	+3.3
Average of 5 runs, 2560	+2.0	+3.5	+2.2	+2.1	–1.0	+0.0	+1.2	–1.4	+2.5	+1.7	–1.1	+2.5	+0.5	+3.7	+3.0	+1.3	+4.1	+8.6
Ensemble of 5 runs, 2560	**+3.3**	**+4.3**	**+3.0**	**+3.0**	+2.3	**+1.3**	**+1.3**	–0.8	**+3.6**	**+2.5**	+1.1	**+3.5**	**+1.8**	**+4.6**	**+3.5**	**+2.3**	**+6.3**	**+11.5**
B) Ensembles for the mT5-xl Model for Various Context Sizes																		
Average of 5 runs, 512	73.3	77.5	78.4	77.2	73.9	76.1	75.4	72.9	80.1	68.4	70.3	67.2	77.2	77.7	78.3	76.1	71.3	47.6
Ensemble of 5 runs, 512	+0.8	+1.1	+0.9	+0.8	–2.3	**+0.2**	+0.8	**+1.9**	+1.1	+1.1	+0.9	+1.8	+1.6	+1.1	+0.8	+1.0	+1.3	+0.3
Average of 5 runs, 768	+1.1	+2.2	+1.3	+1.7	–4.4	+0.1	+1.3	+0.9	+1.7	+1.5	–1.3	+1.9	+1.5	+2.6	+2.2	+0.5	+2.6	+2.4
Average of 5 runs, 2560	+1.9	+3.4	+2.6	+2.6	–4.4	+0.1	+1.7	+1.0	+2.8	+2.0	–1.5	+2.2	+1.2	+3.7	+3.6	+1.4	+5.3	+5.7
Ensemble of 5 runs, 2560	**+3.5**	**+4.9**	**+3.6**	**+3.7**	**+2.4**	**+0.2**	**+2.3**	+1.1	**+3.6**	**+3.3**	**+1.3**	**+4.0**	**+3.0**	**+4.1**	**+5.0**	**+2.5**	**+7.1**	**+7.6**

Table 8: Ablation experiments evaluated on the development sets (CoNLL score in %). We report the average/ensemble of best 5 out of 7 runs, using for every corpus the single epoch achieving the highest average score.

training. Therefore, it could be possible that different mixing ratios influence the best epochs of individual corpora and that with some mixing ratios, the best epochs are more homogeneous. On that account, Table 7.B performs the evaluation differently – for each of the 5 runs, we choose the epoch with the best overall performance on all corpora, and employ the checkpoint from this epoch for all corpora; different runs can utilize different epochs. Nevertheless, the results are very much similar.

5.6 Ensembling

The effect of ensembling the 5 runs (instead of averaging them) is captured in Table 8. For the context size 512, the ensemble delivers an additional 1 percent point with the mT5-large pretrained model and 0.8 percent points with the mT5-xl model. For the context size 2560, the improvement is even slightly larger, 1.3 and 1.6 percent points for the mT5-large and mT5-xl models, respectively.

6 Conclusions

We presented the winning entry to the CRAC 2023 Shared Task on Multilingual Coreference Resolution (Žabokrtský et al., 2023). The system is an improved version of our earlier multilingual coreference pipeline CorPipe (Straka and Straková, 2022), and it surpasses other participants by a large margin of 4.5 percent points. When ensembling is not desired, we also offer a single multilingual checkpoint for all 17 corpora surpassing other submissions by 2.6 percent points. The source code is available at `https://github.com/ufal/crac2023-corpipe`.

Acknowledgements

This work has been supported by the Grant Agency of the Czech Republic, project EXPRO LUSyD (GX20-16819X), and has been using data provided by the LINDAT/CLARIAH-CZ Research Infrastructure (`https://lindat.cz`) of the Ministry of Education, Youth and Sports of the Czech Republic (Project No. LM2023062).

Limitations

The presented system has demonstrated its performance only on a limited set of 12 languages, and heavily depends on a large pretrained model, transitively receiving its limitations and biases.

Furthermore, the practical applicability on plain text inputs depends also on empty node prediction, whose performance has not yet been evaluated.

Training with the mT5-large pretrained model requires a 40GB GPU, which we consider affordable; however, training with the mT5-xl pretrained model needs nearly four times as much GPU memory.

References

Bernd Bohnet, Chris Alberti, and Michael Collins. 2023. Coreference resolution through a seq2seq transition-based system. *Transactions of the Association for Computational Linguistics*, 11:212–226.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking Embedding Coupling in Pre-trained Language Models. In *International Conference on Learning Representations*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–

8451, Online. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. XLM-V: Overcoming the Vocabulary Bottleneck in Multilingual Masked Language Models. *arXiv e-prints*, page arXiv:2301.10472.

Michal Novák, Martin Popel, Zdeněk Žabokrtský, Daniel Zeman, Anna Nedoluzhko, Kutay Acar, Peter Bourgonje, Silvie Cinková, Gülşen Cebiroğlu Eryiğit, Jan Hajič, Christian Hardmeier, Dag Haug, Tollef Jørgensen, Andre Kåsen, Pauline Krielke, Frédéric Landragin, Ekaterina Lapshinova-Koltunski, Petter Mæhlum, M.Antònia Martí, Marie Mikulová, Anders Nøklestad, Maciej Ogrodniczuk, Lilja Øvrelid, Tuğba Pamay Arslan, Marta Recasens, Per Erik Solberg, Manfred Stede, Milan Straka, Svetlana Toldova, Noémi Vadász, Erik Velldal, Veronika Vincze, Amir Zeldes, and Voldemaras Žitkus. 2022. Coreference in Universal Dependencies 1.1 (CorefUD 1.1). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Ondřej Pražák and Miloslav Konopik. 2022. End-to-end multilingual coreference resolution with mention head prediction. In *Proceedings of the CRAC 2022 Shared Task on Multilingual Coreference Resolution*, pages 23–27, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Ondřej Pražák, Miloslav Konopík, and Jakub Sido. 2021. Multilingual coreference resolution with harmonized annotations. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1119–1123, Held Online. INCOMA Ltd.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

Milan Straka and Jana Straková. 2022. ÚFAL CorPipe at CRAC 2022: Effectivity of multilingual models for coreference resolution. In *Proceedings of the CRAC 2022 Shared Task on Multilingual Coreference Resolution*, pages 28–37, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zdeněk Žabokrtský, Miloslav Konopík, Anna Nedoluzhko, Michal Novák, Maciej Ogrodniczuk, Martin Popel, Ondřej Pražák, Jakub Sido, and Daniel Zeman. 2023. Findings of the Second Shared Task on Multilingual Coreference Resolution. In *Proceedings of the CRAC 2023 Shared Task on Multilingual Coreference Resolution*, pages 1–18, Singapore. Association for Computational Linguistics.

Zdeněk Žabokrtský, Miloslav Konopík, Anna Nedoluzhko, Michal Novák, Maciej Ogrodniczuk, Martin Popel, Ondřej Pražák, Jakub Sido, Daniel Zeman, and Yilun Zhu. 2022. Findings of the shared task on multilingual coreference resolution. In *Proceedings of the CRAC 2022 Shared Task on Multilingual Coreference Resolution*, pages 1–17, Gyeongju, Republic of Korea. Association for Computational Linguistics.