Accelerating Cross-Encoders in Biomedical Entity Linking

Javier Sanz-Cruzado
University of Glasgow
Glasgow, United Kingdom
javier.sanz-cruzadopuig@glasgow.ac.uk

Jake Lever
University of Glasgow
Glasgow, United Kingdom
jake.lever@glasgow.ac.uk

Abstract

Biomedical entity linking models disambiguate mentions in text by matching them with unique biomedical concepts. This problem is commonly addressed using a two-stage pipeline comprising an inexpensive candidate generator, which filters a subset of suitable entities for a mention, and a costly but precise reranker that provides the final matching between the mention and the concept. With the goal of applying two-stage entity linking at scale, we explore the construction of effective cross-encoder reranker models, capable of scoring multiple mention-entity pairs simultaneously. Through experiments on four entity linking datasets, we show that our cross-encoder models provide between 2.7 to 36.97 times faster training speeds and 3.42 to 26.47 times faster inference speeds than a base cross-encoder model capable of scoring only one entity, while achieving similar accuracy (differences between -3.42% to 2.76% Acc@1).

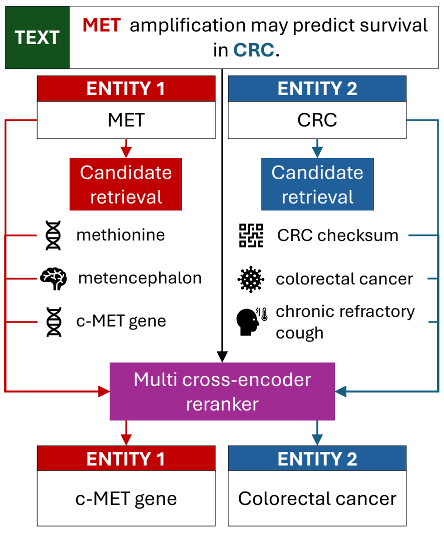


Figure 1: Example of our multi cross-encoder model.

1 Introduction

Biomedical entity linking matches mentions of biomedical concepts (diseases, chemicals) in texts with unique entities within a knowledge base (Kartchner et al., 2023; Garda et al., 2023). Disambiguating mentions within text is fundamental for information extraction tasks, as a single entity might be referred to by different names or aliases (e.g. chickenpox and varicella refer to the same disease), or a mention might refer to multiple entities (e.g. APC might refer to advanced pancreatic or prostate cancer).

This problem is commonly devised as a two-phase procedure (Xu et al., 2023): given a mention in a text, an initial model selects a reduced set of candidate entities it might refer to. This model is commonly fast, as it needs to filter among thousands of entities (Neumann et al., 2019; Liu et al., 2021). Then, a second, more precise model reranks these candidate entities to provide the final matching between a mention and an entity in the knowledge-base. Cross-encoders (Logeswaran et al., 2019) are a popular option for this stage (Wu et al., 2020; Zhang et al., 2022). While accelerated by the reduced candidate selection, these rerankers are usually costly, requiring extensive training and inference times. The efficiency of these rerankers can be an important factor of our entity linking pipeline if we want to run these models at scale across millions of documents. However, works on biomedical entity linking have traditionally focused on the accuracy of the models and not on their efficiency.

Therefore, in this paper, we aim to improve the efficiency of second stage cross-encoder models. Taking as a starting point a base cross-encoder (Logeswaran et al., 2019), we propose novel entity linking methods that improve both training and inference speeds, while maintaining similar accuracy levels. For this, inspired by the longer context windows of recent encoder-only transformer models like ModernBERT (Warner et al., 2024), we design cross-encoders capable of scoring multiple candidates at the same time, and even reranking multiple mentions simultaneously. We show an example of this in Figure 1. Our contributions are threefold:

- We propose a novel multi cross-encoder architecture that accelerates the training and inference times of a classical reranker cross-encoder for entity linking.
- We compare our approach on three different transformer models and four different biomedical entity linking datasets.
- We find that our cross-encoders can accelerate up to 36.97 times the training speed of a simple cross-encoder and up to 26.47 times the inference speed, while providing similar effectiveness.

2 Task definition and notation

We start by formally defining the entity linking (EL) task. EL aims to uniquely match entities mentioned in the text with unique concepts within a knowledge base. Let's suppose we have a knowledge base containing a set of unique entities $\mathcal{E}$ and a corpus of documents $\mathcal{D}$. Each document $d \in \mathcal{D}$ has a series of mentions $\mathcal{M}_d$, where a mention $m \in \mathcal{M}_d$ is a sequence of tokens $m = d_m^{(1)} \cdots d_m^{(l)} \subseteq d$ that corresponds to a unique entity. Given a document d and a mention $m \in \mathcal{M}_d$, the EL task consists on identifying the entity $e_m \in \mathcal{E}$ that m refers to in d. We address this task as a ranking problem, where we find the entity maximizing a ranking function $f_{m,d} : \mathcal{E} \to \mathbb{R}$.

3 Related work

While early works on biomedical entity linking date back to the late 1980s (French and McInnes, 2023), a majority of recent works in this area are based on recent transformer-based language models like BERT (Devlin et al., 2019) or BiomedBERT (Gu et al., 2021). These works can be divided into single-phase and two-phase models.

Single-phase models directly rank all entities within a knowledge-base for a single mention. These methods usually estimate the similarity between mentions and entities based on a combination of sparse or dense vector representations (Sung et al., 2020; Loureiro and Jorge, 2020). These models commonly use computationally efficient algorithms like bi-encoders to obtain separate representations of entities and mentions. Examples of algorithms within this category are SapBERT (Liu et al., 2021), BioSyn (Sung et al., 2020) and MedLinker (Loureiro and Jorge, 2020).

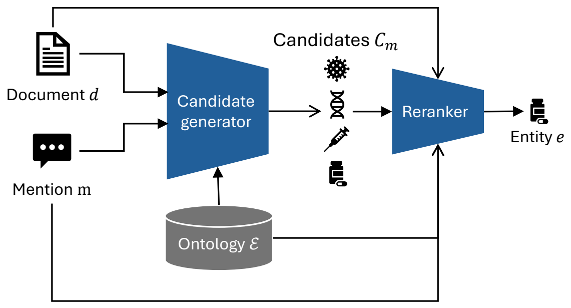


Figure 2: Two-stage pipeline

This work focuses on the second type of models, the two-phase models. These algorithms apply two different entity linking approaches: a fast and efficient model for retrieving a subset of candidates (for instance, a character n-grams model (Angell et al., 2021) or a bi-encoder like SapBERT (Xu et al., 2023; Zhu et al., 2024)), followed by a more computationally expensive, but precise reranker that reranks the set of candidate items. Following Logeswaran et al. (2019), a majority of these models use a cross-encoder model as a reranker. While different biomedical EL models like ClusterEL (Angell et al., 2021), ArboEL (Agarwal et al., 2022), KrissBERT (Zhang et al., 2022) apply a similar pipeline, all of these models have focused their attention only on the effectiveness of the entity linking model, commonly using a simple cross-encoder model for reranking. However, there is still room to improve not only the effectiveness, but also the efficiency of these approaches.

To improve the efficiency of these models, we get inspiration from the Prompt-BioEL method proposed by (Xu et al., 2023). In their approach, they apply a cross-encoder capable of processing all the candidates for a mention simultaneously. In this work, we go further, by building cross-encoder models which can rerank multiple mentions, sentences or passages at the same time.

4 Method

In this section, we describe our approach for balancing the accuracy and efficiency of entity linking models. Figure 2 illustrates the general architecture of our entity linking models. Following previous works (Zhang et al., 2022; Logeswaran et al., 2019), we adopt a two-stage pipeline for the task. First, we apply a **candidate retrieval** model, which selects a small subset of candidate entities $C_m \subset \mathcal{E}$ from the ontology. Then, we apply a **reranker** model that chooses the best entity among the ones in C_m. We next describe each of these components.

4.1 Candidate retriever

As an efficient and effective first-stage candidate retriever, we use a n-grams model (Neumann et al., 2019) for representing both the mention text m and the aliases of entity e (which we denote as $A(e)$). Then, we rank candidate entities by the maximum TF-IDF similarity between m and every alias $a_e \in A(e)$ of the entity.

$$f_{m,d}(e) = \max_{a_e \in A(e)} \text{tf-idf}(\text{n-gr}(m),\ \text{n-gr}(a_e)) \quad (1)$$

where n-gr(t) is the n-grams representation of t.

4.2 Reranker

As a second-stage candidate reranker, we use cross-encoder models. Cross-encoders have been previously used for the entity linking task, but they are costly to train and apply. Therefore, we propose improvements over the basic cross-encoder architecture, illustrated in Figure 3.

4.2.1 Preprocessing

As an initial step, prior to the application of the reranker, we pre-process the documents in our corpus. We divide the documents into passages, and each passage into sentences. Each annotated sentence is later provided as input to the cross-encoder models, providing context for each mention.

4.2.2 Base cross-encoder

We first describe the architecture of the base cross-encoder model (Humeau et al., 2020), depicted in Figure 3(a). Given an annotated sentence $t \subseteq d$, a mention m and a candidate entity $c \in C_m$ for that mention, the cross-encoder computes a score $f_{m,d}(c)$ estimating the likelihood that the candidate entity c corresponds to the target entity e_m that mention m is referring to. Each candidate $c \in C_m$ is processed separately, and then, candidates are ranked in descending score order.

A common strategy to build the cross-encoder (and the one we follow in this work) is to fine-tune a pre-trained language model (LM). The LM receives as input a sentence following the following template $\tau(t, m, c)$[1]:

$$\tau(t, m, c) = \text{“}t \text{ [SEP] } m \text{ [MASK] } c\text{”} \quad (2)$$

[1]For our cross-encoder models, we represent c as the main textual representation of the entity in the knowledge base

An example of an input sentence is shown in Figure 4. Then, the cross-encoder classifies the [MASK] token into two classes: a positive class, indicating that c matches the referred entity e_m, and a negative class otherwise. Therefore, the score $f_{m,d}(c)$ is defined as:

$$f_{m,d}(c) = p\left(\text{[MASK]} = 1 | \tau\left(t, m, c\right)\right) \quad (3)$$

In order to fine-tune the model, we apply a cross-entropy loss minimizing the classification error on the [MASK] token. For a candidate c and a mention m, the loss is defined as

$$\begin{aligned}\mathcal{L} = &-\mathbb{1}\left(c = e_m\right) \cdot \log f_{m,d}(c) \\ &- \left(1 - \mathbb{1}\left(c = e_m\right)\right) \cdot \log\left(1 - f_{m,d}(c)\right)\end{aligned} \quad (4)$$

where $\mathbb{1}(x)$ is the indicator function.

4.2.3 Parallel cross-encoder

One of the limitations of the architecture of the base cross-encoder is its capacity to process only one mention-candidate pair at a time. Therefore, in both training and inference, the cross-encoder needs to process the sentence t as many times as candidates we retrieve during the first phase of the entity linking process – making this process costly.

Inspired by Xu et al. (2023) and Jiang et al. (2023), we propose to improve our cross-encoder by allowing it to process multiple candidates for a mention at the same time. We denote the new architecture as parallel cross-encoder. As illustrated in Figure 3(b), the parallel cross-encoder takes as input the text t and all the candidates for mention m, and provides, as output, the scores for all of those candidates. The parallel cross-encoder receives input with the following template:

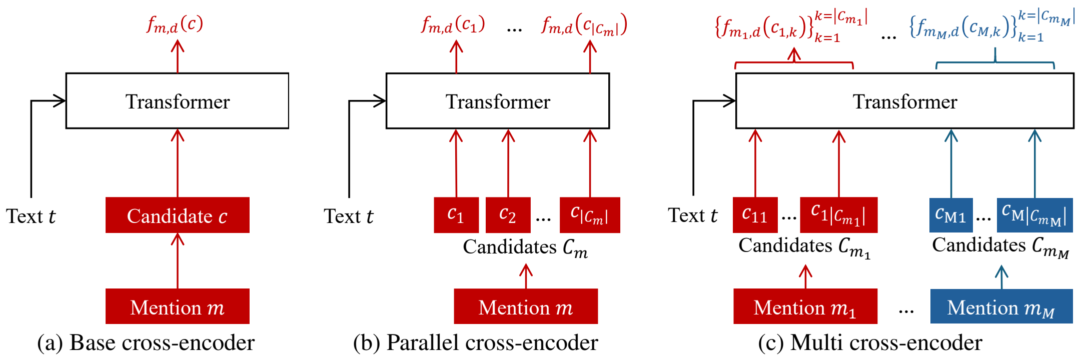
$$\begin{aligned}\tau(t, m) = \text{“}t \text{ [SEP] } m \text{ [MASK] } c_1 \\ \cdots \\ \text{[SEP] } m \text{ [MASK] } c_{|C_m|}\text{”}\end{aligned} \quad (5)$$

and, for each candidate $c \in C_m$, its score is

$$f_{m,d}(c) = p\left(\text{[MASK]}_c = 1 | \tau\left(t, m\right)\right) \quad (6)$$

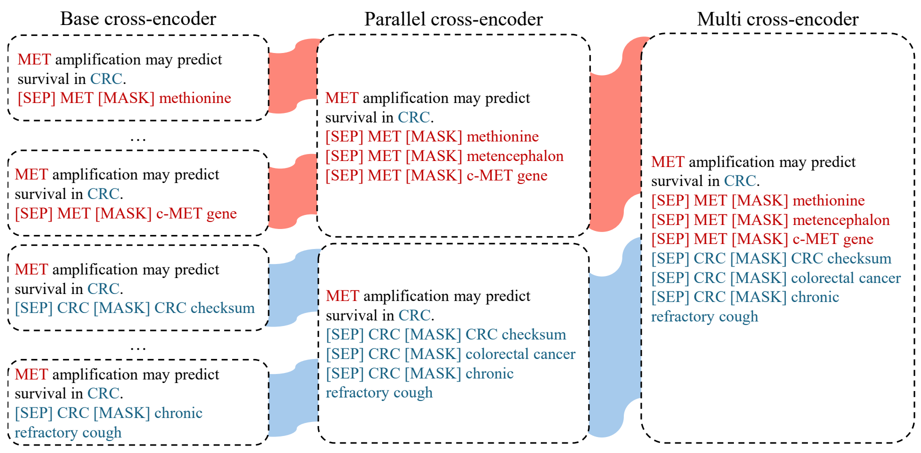
where [MASK]_c is the mask token corresponding to entity c. An example of this input text is shown in Figure 4.

While the parallel cross-encoder increases the complexity of the task (the cross-encoder receives longer text sequences and needs to classify multiple tokens), it should accelerate training and inference times. As long as the cross-encoder effectively processes long sequences of tokens, we should gain advantage from processing sentence t only once for a given mention.



$f_{m,d}(c)$
Transformer
Text t
Candidate c
Mention m
(a) Base cross-encoder
$f_{m,d}(c_1)$ ... $f_{m,d}(c_{|C_m|})$
Transformer
Text t
c_1 c_2 ... $c_{|C_m|}$
Candidates C_m
Mention m
(b) Parallel cross-encoder
$\{f_{m_1,d}(c_{1,k})\}_{k=1}^{k=|C_{m_1}|}$ ... $\{f_{m_M,d}(c_{M,k})\}_{k=1}^{k=|C_{m_M}|}$
Transformer
Text t
c_{11} ... $c_{1|C_{m_1}|}$
Candidates C_{m_1}
c_{M1} ... $c_{M|C_{m_M}|}$
Candidates C_{m_M}
Mention m_1 ... Mention m_M
(c) Multi cross-encoder

Figure 3: Architecture of the different cross-encoder models



Base cross-encoder
MET amplification may predict survival in CRC. [SEP] MET [MASK] methionine
...
MET amplification may predict survival in CRC. [SEP] MET [MASK] c-MET gene
MET amplification may predict survival in CRC. [SEP] CRC [MASK] CRC checksum
...
MET amplification may predict survival in CRC. [SEP] CRC [MASK] chronic refractory cough
Parallel cross-encoder
MET amplification may predict survival in CRC. [SEP] MET [MASK] methionine [SEP] MET [MASK] metencephalon [SEP] MET [MASK] c-MET gene
MET amplification may predict survival in CRC. [SEP] CRC [MASK] CRC checksum [SEP] CRC [MASK] colorectal cancer [SEP] CRC [MASK] chronic refractory cough
Multi cross-encoder
MET amplification may predict survival in CRC. [SEP] MET [MASK] methionine [SEP] MET [MASK] metencephalon [SEP] MET [MASK] c-MET gene [SEP] CRC [MASK] CRC checksum [SEP] CRC [MASK] colorectal cancer [SEP] CRC [MASK] chronic refractory cough

Figure 4: Input of the cross-encoder models.

4.2.4 Multi cross-encoder

The parallel cross-encoder can be further refined to improve the efficiency of the cross-encoder by reducing the amount of times that the cross-encoder is exposed to the same text. As shown in Figure 1, each sentence might contain not only one, but multiple mentions to entities in the knowledge base. Therefore, we propose a new cross-encoder model, denoted as multi cross-encoder that receives as input not only the candidates of an individual mention, but the candidates of all mentions within the sentence and provides the corresponding estimates. We illustrate this architecture in Figure 3(c).

The multi cross-encoder works similarly to the parallel cross-encoder. If we denote as $\mathcal{M}_t \subseteq \mathcal{M}_d$ the set of mentions in a sentence t, and $|\mathcal{M}_t| = M$, we define the input text of the multi-cross encoder as a sequence of tokens with the following format:

$$\begin{aligned}\tau\left(t, \mathcal{M}_t\right) = \text{“}t\ &[\text{SEP}]\ m_1\ [\text{MASK}]\ c_{1,1}\\ &\dots\\ &[\text{SEP}]\ m_1\ [\text{MASK}]\ c_{1,|C_{m_1}|}\\ \\ &[\text{SEP}]\ m_2\ [\text{MASK}]\ c_{2,1}\\ &\dots\\ &[\text{SEP}]\ m_M\ [\text{MASK}]\ c_{M,|C_{m_M}|}\text{”}\end{aligned} \tag{7}$$

We provide an example on Figure 4. Then, the score for a candidate $c \in C_m$ is defined as:

$$f_{m,d}(c) = p\left([\text{MASK}]_{m,c} = 1 | \tau\left(t, \mathcal{M}_t\right)\right) \tag{8}$$

where $[\text{MASK}]_{m,c}$ is the mask token corresponding to mention m and candidate $c \in C_m$ in $\tau(t, \mathcal{M}_t)$.

4.2.5 Adaptation to context window

As we concatenate multiple mention-entity pairs in the input text, we might obtain texts longer than the context window of the language model (maximum number of tokens that the LM can receive at once). In that case, we partition the mention-candidate pairs into several rankings by choosing, each time, as many pairs as we can fit along the sentence into the context window. We apply this strategy as our method provides pointwise scores (each mention-candidate pair has an individual score) – and therefore, separating the pairs on different calls to the cross-encoder should not have a big impact on performance. In the worst case, where only one mention-entity pair fits in the context, our model would be equivalent to the base cross-encoder. We show an example of this procedure in Figure 5. Following this procedure, LMs with longer context

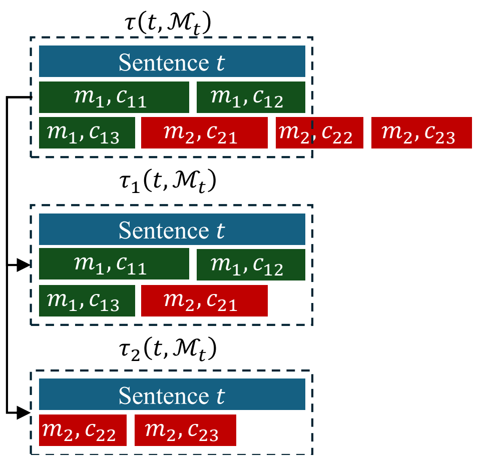


Figure 5: Partition procedure when an input sentence is longer than the context window.

Table 1: Dataset properties.

Property	MedMentions	NCBI Disease	NLM Chem	BC5CDR
Ontology	UMLS	Medic	Mesh 2021	Mesh 2015
Documents (train)	2,635	593	80	500
Documents (val)	878	100	20	500
Documents (test)	879	100	50	500
Passages (train)	2,635	593	5,555	1,000
Passages (val)	878	100	1,285	1,000
Passages (test)	879	100	3,470	1,000
Sentences (train)	25,836	5,173	20,126	4,242
Sentences (val)	8,508	888	4,855	4,299
Sentences (test)	8,597	901	12,031	4,524
Entities (train)	211,029	4,836	19,361	9,323
Entities (val)	71,062	711	4,927	9,570
Entities (test)	70,405	896	11,164	9,725

windows (like ModernBERT (Warner et al., 2024)) shall have longer input texts than models accepting less tokens (like BiomedBERT (Gu et al., 2021)).

4.2.6 Further architectures

We can further design additional cross-encoder architectures that include even more data – with the objective of maximizing the use of the cross-encoder context window – by concatenating the template $\tau(t, \mathcal{M}_t)$ of multiple sentences within the document (for instance, all sentences within a passage or all sentences within a document).

5 Experimental setup

5.1 Datasets

In our experiments, we consider four common datasets for biomedical entity linking:

- **MedMentions (Mohan and Li, 2019):** PubMed biomedical abstract collection annotated with mentions of entities in the UMLS 2017AA release. We use the full version of the dataset. For each entity, we only keep the English aliases.

- **NCBI Disease (Doğan et al., 2014):** PubMed abstract corpus linking disease mentions to entities in the MEDIC ontology[2]. Only mentions with an unambiguous entity link with an entity in that MEDIC release were kept.

- **NLM Chem (Islamaj et al., 2021):** Set of full-text articles from the PubMed Central Open Access dataset covering the use of chemical names in the biomedical literature. We keep only mentions of type 'Chemical' linked with entities in the MeSH 2021 release.

- **BioCreative V CDR (BC5CDR) (Li et al., 2016):** Collection of PubMed abstracts with chemical and disease annotations from the Comparative Toxicogenomics Database. Only contiguous mentions were kept and all linked entities are found in the MeSH 2015 release.

Data splitting: For each of the four datasets, we use the default training/validation/test split. We use the training and validation datasets to fine-tune the models, and we report entity linking results over the test set.

Passages and sentences: For the MedMentions and NCBI Disease datasets, each document consists of a single passage combining both the title and abstract. In the BC5CDR dataset, we have two passages for each dataset: one for the title, and another one for the abstract text. Finally, for NLM Chem, we use the passage division of each document provided by the dataset. For splitting each passage into sentences, we use the spaCy[3] en_core_web_sm sentence parser. If the parser splits a mention in two different sentences, we combine the two sentences. We show the statistics of each dataset in Table 1.

5.2 Models

First-phase candidate retriever: As mentioned in Section 4.1, we use a TF-IDF n-grams model (Neumann et al., 2019). We apply an efficient implementation of this model by building an n-grams index with Pyterrier-PISA (Mallia et al., 2019; MacAvaney and Macdonald, 2022). For efficiency, this index is built only using the first 16 characters of entity aliases. Then, for each dataset, we use the n-grams model maximizing the amount of correct

[2] 10 May 2012 version, obtained using Internet Archive

[3] spaCy: `https://spacy.io/`

Table 2: Language model statistics

Model	Domain	Context-window length
BiomedBERT	Biomedical	512
Longformer	General	4,096
ModernBERT	General	8,192

entities in the top-5 (3-grams for MedMentions, 2-grams for the rest).

Second-phase reranker models: Then, we build cross-encoders for reranking the top-5 candidate entities. We consider three different backbone pre-trained language models in our experiments, with varying context window size: BiomedBERT (Gu et al., 2021), Longformer (Beltagy et al., 2020) and ModernBERT (Warner et al., 2024). Table 2 summarizes their statistics.

As a baseline for our experiments, we consider the base cross-encoder defined in Section 4.2.2 (which we denote as LM-base). We compare this baseline against four models: the parallel cross-encoder (LM-parallel) and the multi cross-encoder (LM-multi) in Sections 4.2.3 and 4.2.4 , and two additional cross-encoders: one including the whole passage text (LM-passage), and another one including the complete document text (LM-document).

To reduce the training time of each model, all cross-encoders follow an early stopping strategy, where we stop the training if the cross-encoder fails to improve the F1 performance on the validation set by 1% for three consecutive epochs. All models use the same learning rate (10^{-6}).

5.3 Metrics

We compare our models across three main metrics:

- **Accuracy@1 (Acc@1):** This metric measures the ultimate goal of the reranker to assign the highest score to the correct entity for each mention from the list of candidates. It is the proportion of annotations for which this is the case.

- **Training speed:** This metric measures the efficiency of the fine-tuning process. As it is unfair to compare models directly on the training time (as different models might use a different number of training epochs), we estimate the number of annotations processed per second during the cross-encoder fine-tuning.

- **Inference speed:** This metric estimates the number of test examples per second that the cross-encoder can process.

For reference, we also report the total training and inference times of our cross-encoder models.

5.4 Hardware

We train and execute all our models on a single NVIDIA RTX 4090 GPU card (24 GB VRAM), 2 CPUs and 16 GB of RAM. The batch size of each model has been adjusted to be trained on the mentioned GPU card – with all variations of the same model using the same batch size.

5.5 Implementation

For reproducibility, we provide the code for our experiments in the following GitHub repository: `https://github.com/Glasgow-AI4BioMed/entitytools`.

6 Results

We aim to answer the following research questions:

- **RQ1:** How does the parallelism of the cross-encoder affect the effectiveness of the model?

- **RQ2:** How does the parallelism of the cross-encoder affect the model training and inference speeds?

6.1 RQ1: Accuracy comparison

We first analyse how effective the different cross-encoders are for the entity linking task. As we keep adding more information to our cross-encoder, we shall expect the task to become more complex and therefore affect the effectiveness of our models. We show the results in Tables 3 (for the Medmentions and NCBI disease datasets) and 4 (for the NLM Chem and BC5CDR corpora). In these tables, we underline the best result for each metric and backbone LM for our cross-encoders, and we highlight in bold the best overall result. Accuracy@1 results are shown in the first column for each dataset.

Base cross-encoder performance: We evaluate the effectiveness of the two-stage entity linking model by comparing the base cross-encoders with the single-stage n-grams model. In line with other works (Zhang et al., 2022; Agarwal et al., 2022), the three base cross-encoders achieve statistically significant improvements (McNemar test with $p < 0.05$ and Bonferroni correction) on a majority of datasets. The only exception is the NLM Chem dataset, where only the domain-specific BiomedBERT cross-encoder can improve the n-grams similarity model. Among the three

Table 3: Evaluation of entity linking (Medmentions and NCBI Disease). For each model, ^{a} represents statistical significance (McNemar test with Bonferroni correction and $p < 0.05$) with respect to the first stage linker. ^{b,c,d,e} represent, respectively, a significant improvement over the simple, parallel, multi or passage cross-encoder with the same base transformers model. For each metric, ↑ indicates that higher values are better, while ↓ indicates that lower values are better. Best values are highlighted in bold, and the best cross-encoder for each backbone LM is underlined.

	Medmentions-full					NCBI Disease				
Model	Acc@1 (↑)	Training speed (↑)	Training time (s) (↓)	Inference speed (↑)	Inference time (s) (↓)	Acc@1 (↑)	Training speed (↑)	Training time (s) (↓)	Inference speed (↑)	Inference time (s) (↓)
First stage	0.4222	202.52	**1,042**	1,466.77	44	0.5960	322.40	**15**	**896.00**	1
BiomedBERT-base	**0.6422**^{acde}	62.05	23,806	145.46	484	0.6886^{a}	70.09	690	179.20	5
BiomedBERT-parallel	0.6334^{ade}	255.74	4,951	690.25	102	0.6931^{a}	266.55	127	**896.00**	**1**
BiomedBERT-multi	0.6244^{a}	639.16	1,981	2,200.16	32	0.6864^{a}	318.16	152	**896.00**	**1**
BiomedBERT-passage	0.6235^{a}	**985.46**	1,499	**2,607.59**	**27**	0.6786^{a}	**531.43**	91	**896.00**	**1**
Longformer-base	0.6353^{acde}	4.44	190,014	31.64	2,225	0.6920^{a}	5.00	7,744	37.33	24
Longformer-parallel	0.6284^{ade}	21.55	39,162	158.93	443	0.6920^{a}	22.44	862	179.20	5
Longformer-multi	0.6203^{ae}	137.40	9,215	733.39	96	0.6953^{a}	54.54	798	298.67	3
Longformer-passage	0.6136^{a}	168.69	8,757	869.20	81	0.6842^{a}	81.97	531	448.00	**2**
ModernBERT-base	0.6300^{ade}	11.84	71,305	44.99	1,440	0.6875^{a}	12.94	1,869	48.89	20
ModernBERT-parallel	0.6287^{ad}	58.03	14,546	237.05	297	0.6953^{a}	59.41	407	179.20	5
ModernBERT-multi	0.6255^{a}	205.88	4,100	1117.54	63	**0.7042**^{a}	57.92	501	298.67	3
ModernBERT-passage	0.6266^{a}	196.90	8,574	529.36	133	0.7009^{a}	159.68	212	448.00	**2**

Table 4: Evaluation of entity linking (NLM Chem and BC5CDR). For each model, ^{a} represents statistical significance (McNemar test with Bonferroni correction and $p < 0.05$) with respect to the first stage linker. ^{b,c,d,e,f} represent, respectively, a significant improvement over the simple, parallel, multi, passage or document cross-encoder with the same base transformers model. For each metric, ↑ indicates that higher values are better, while ↓ indicates that lower values are better. Best values are highlighted in bold, and the best cross-encoder for each backbone LM is underlined.

	NLM Chem					BC5CDR				
Model	Acc@1 (↑)	Training speed (↑)	Training time (s) (↓)	Inference speed (↑)	Inference time (s) (↓)	Acc@1 (↑)	Training speed (↑)	Training time (s) (↓)	Inference speed (↑)	Inference time (s) (↓)
First stage	0.6990	108.77	178	**1594.86**	7	0.8193	61.34	152	1389.29	7
BiomedBERT-base	0.7004	54.34	2,494	150.86	74	0.8403^{adef}	58.64	1,113	149.62	65
BiomedBERT-parallel	**0.7083**^{abf}	211.60	366	697.75	16	**0.8444**^{adef}	216.81	301	694.64	14
BiomedBERT-multi	0.7066^{a}	268.90	288	1014.91	11	0.8249^{a}	310.77	120	1389.29	7
BiomedBERT-passage	0.7075^{ab}	378.39	307	1240.44	9	0.8274^{a}	443.95	168	**1620.83**	**6**
BiomedBERT-document	0.7043^{a}	**556.35**	**174**	**1594.86**	**7**	0.8254^{a}	**570.80**	**98**	**1620.83**	**6**
Longformer-base	0.6940	4.59	16,864	33.83	330	0.8335^{aef}	4.02	16,242	33.89	287
Longformer-parallel	0.7068^{abef}	22.18	3,492	164.18	68	0.8427^{abef}	18.71	2,492	159.43	61
Longformer-multi	0.7044^{abef}	57.97	1,336	310.11	36	0.8414^{abef}	62.15	900	374.04	61
Longformer-passage	0.7014^{b}	68.60	1,129	413.48	27	0.8209^{f}	73.85	505	422.83	23
Longformer-document	0.7018^{b}	115.24	1,176	656.71	17	0.8188	94.41	395	607.81	16
ModernBERT-base	0.6882	12.07	6,416	45.75	244	0.8311^{a}	13.02	3,581	51.18	190
ModernBERT-parallel	0.7072^{abf}	57.32	1,351	232.58	48	0.8431^{abdef}	47.87	779	226.16	43
ModernBERT-multi	0.7050^{abf}	60.46	2,562	446.56	25	0.8393^{abf}	58.45	638	237.20	41
ModernBERT-passage	0.7066^{abf}	129.07	1,050	558.20	20	0.8357^{a}	140.90	397	221.02	44
ModernBERT-document	0.6972	74.61	2,076	218.90	51	0.8344^{a}	174.49	374	607.81	16

language models, BiomedBERT achieves the best results as a base encoder across the four datasets – thus highlighting the advantage of a language model pre-trained on domain-specific text.

Multi cross-encoder performances: We then analyse the effect of adding more information to the cross-encoders. In general, increasing the number of mentions fed to the model in one input does degrade predictive performance, but only minorly. The extreme cases of passing in a whole passage (MedMentions/NCBI Disease) or whole document (NLMChem/BC5CDR) generally achieve the worst accuracy results. However, consistent with the work by Xu et al. (2023), in some of the datasets, adding some mention-entity pairs to classify can provide an improved context and help the accuracy. This is the case of the parallel cross-encoders in NCBI Disease, NLM Chem and BC5CDR and the Longformer and ModernBERT multi-cross encoders in NCBI Disease.

Notably, accuracy results between variants of the same model are commonly small – ranging between -3.42% accuracy decrease to 2.76% performance increase with respect to the base cross-encoder. This, along with the fact that even the worst cross-encoder variants are commonly significantly better than the first-stage linker, makes the different cross-encoder variants reasonable algorithms for the biomedical entity linking task.

To answer RQ1: In general, providing more mention-entity pairs as input to cross-encoder models has limited impact on the Acc@1 performance (ranging between -3.42% to 2.76%). Different datasets can react differently to the amount of information provided to the cross-encoder, so it is important to choose the right model to enhance the accuracy. However, the small accuracy differences make all the variants reasonable entity linkers.

6.2 RQ2: Efficiency comparison

We now analyse the training and inference speeds for each of our tested datasets. Results are shown in the 2nd and 4th columns for each dataset in Tables 3 and 4. As different models use different batch sizes to fit the model into a single GPU card, we shall only compare speeds across encoders with the same base LM.

For completion, Tables 3 and 4 include the inference times and speeds of the first stage n-gram models. However, it must be noted that the n-grams model has differences with respect to the cross-encoders. Specifically, first-stage processing is not accelerated by GPU, is not trained for multiple epochs and does not depend on batching. Therefore, the training and inference speeds are not directly comparable to those of cross-encoder models. However, we include them as a reference of how fast it is to train and apply each of these models.

Training speed: Our results show a clear trend where the base cross-encoders are the slowest second stage models in our comparison, and the training speed generally increases as we enhance the parallelism of the cross-encoders. The parallel cross-encoders increase the training speed of the baseline models between 2.68 (13.02 examples/s to 47.87 examples/s for the ModernBERT model on BC5CDR) and 3.9 times (11.84 examples/s to 58.53 examples/s for ModernBERT on MedMentions). Models with more examples reach even higher speeds, with multi cross-encoders achieving 3.49-29.93 speed improvements, and passage and document models achieving even further gains.

While increasing the number of mention-entity pairs commonly increases the training speed, that does not always occur. We observe two exceptions in our experiments: ModernBERT-passage in MedMentions and ModernBERT-document in NLM Chem. We hypothesize that the advantage of adding more pairs to the input text depends on the capacity of the cross-encoder of processing those examples. As we add more tokens to the input text, the cross-encoder might reach a point where it slows down its processing. We show this in Figure 6. In this figure, we illustrate the average input token length of the training examples of each model (x-axis) against the training speed (y-axis). Each line represents a backbone LM, with the arrow indicating the model receiving more information per example. As we can observe here, the only case where the speed decreases (with respect to the previous model) is the ModernBERT-document model, where inputs reach 6,000 tokens on average (2000 more than the second with the longer text sequence, Longformer-document). A similar observation occurs on Medmentions.

Inference speed: When it comes to inference time, we observe similar patterns with respect to the training time, with the base cross-encoders being the slowest models, and the models scoring multiple mention-entity pairs at the same time achieving speed improvements between 3.42 times (ModernBERT-parallel on BC5CDR from 51.18 examples/s to 226.16 examples/s) to 26.47 times (Longformer-passage from 31.64 examples/s to 869.2 examples/s on MedMentions) with respect to the base model[4]. Again, some slowdown is observed when the input token length notably surpasses the size of the Longformer window (as in ModernBERT-passage for MedMentions and ModernBERT-document for NLM Chem).

To answer RQ2: Allowing cross-encoders to simultaneously score multiple entities notably boosts the training and inference speeds of entity linking models. However, models with larger context windows might face difficulties when the length of the input text is too long – effectively reducing the efficiency improvements provided by these models.

7 Conclusions

In this work, we have studied the use of advanced cross-encoder models as rerankers for an entity linking pipeline. These advanced cross-encoders enable simultaneously processing of several mention-candidate pairs, accelerating the training speed of a base cross-encoder by a factor between 2.68 and 36.97, and the inference speed by a

[4]We observe a few ties between rerankers on NCBI Disease and BC5CDR. These are due to the small inference times of these models on each dataset (<10 s).

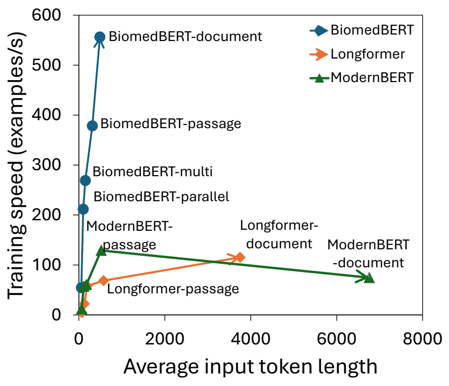


Figure 6: Input token length vs. training speed (NLM Chem)

factor between 3.42 and 26.47. While the parallel processing of multiple candidates might hurt the performance of the model, we find this effect to be small (up to 3.42% accuracy loss). Speed advantages, along with the low accuracy losses make these architectures suitable for environments where training and inference speed is crucial (like real time services).

This work has only focused on cross-encoder with point-wise losses, where we directly estimate the probability of a candidate for a mention. As future work, we shall explore the effect of this cross-encoder architecture on pair-wise or list-wise cross-encoder rerankers, considering the order between pairs. Furthermore, we propose the application of similar architectures on other second stage rerankers, like bi-encoders or poly-encoders (Humeau et al., 2020), that encode mentions and candidates separately.

Limitations

The effectiveness and efficiency of our approach are influenced by two factors: (a) the base model selection and (b) the dataset on which the cross-encoders are trained and applied.

Model selection: In this paper, we only focus on a simple entity linking pipeline, based on an n-grams TF-IDF model for candidate retrieval (Neumann et al., 2019) and variations of the simple cross-encoder reranker described in Logeswaran et al. (2019) and Wu et al. (2020). Although several improvements for the first and second stages of the entity linking pipeline have been developed (Zhu et al., 2024; Angell et al., 2021; Agarwal et al., 2022), we have not tested them in our experiments. However, as most of these proposals use a cross-encoder as their reranker, we believe that similar results should be consistent with our findings if we modified the cross-encoders of these methods in a similar manner.

Dataset selection: In our experiments, we have tested our cross-encoder models on four datasets. The biggest dataset in our experiments is the MedMentions dataset, with only 4,392 documents and 352,496 mentions. While there are bigger biomedical entity linking datasets available, like WikiMed or PubmedDS (Vashishth et al., 2021), training some of the base cross-encoders on MedMentions already represents a challenge (the Longformer-base model takes more than 2 days to train on a single GPU card). By testing our approach across these 4 (smaller) datasets, we show the generalizability of our approach and how it would perform on those bigger datasets.

Acknowledgements

The work is supported by the U.S. National Cancer Institute (NCI) [U24CA275783].

References

Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. 2022. Entity Linking via Explicit Mention-Mention Coreference Modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2022)*, pages 4644–4658, Seattle, WA, USA. Association for Computational Linguistics.

Rico Angell, Nicholas Monath, Sunil Mohan, Nishant Yadav, and Andrew McCallum. 2021. Clustering-based Inference for Biomedical Entity Linking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2021)*, pages 2598–2608, Online. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *Preprint*, arXiv:2004.05150.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019) Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, MN, USA. Association for Computational Linguistics.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10.

Evan French and Bridget T. McInnes. 2023. An overview of biomedical entity linking throughout the years. *Journal of Biomedical Informatics*, 137:104252.

Samuele Garda, Leon Weber-Genzel, Robert Martin, and Ulf Leser. 2023. BELB: a biomedical entity linking benchmark. *Bioinformatics*, 39(11):btad698.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare*, 3(1).

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, Online.

Rezarta Islamaj, Robert Leaman, Sun Kim, Dongseop Kwon, Chih-Hsuan Wei, Donald C. Comeau, Yifan Peng, David Cissel, Cathleen Coss, Carol Fisher, Rob Guzman, Preeti Gokal Kochar, Stella Koppel, Dorothy Trinh, Keiko Sekiya, Janice Ward, Deborah Whitman, Susan Schmidt, and Zhiyong Lu. 2021. NLM-Chem, a new resource for chemical entity recognition in PubMed full text literature. *Scientific Data*, 8:91:1–91:12.

Chengyue Jiang, Wenyang Hui, Yong Jiang, Xiaobin Wang, Pengjun Xie, and Kewei Tu. 2023. Recall, Expand, and Multi-Candidate Cross-Encode: Fast and Accurate Ultra-Fine Entity Typing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023) Volume 1 (Long Papers)*, pages 11597–11609, Toronto, Canada. Association for Computational Linguistics.

David Kartchner, Jennifer Deng, Shubham Lohiya, Tejasri Kopparthi, Prasanth Bathala, Daniel Domingo-Fernández, and Cassie Mitchell. 2023. A Comprehensive Evaluation of Biomedical Entity Linking Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, pages 14462–14478, Singapore. Association for Computational Linguistics.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016:baw068.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2021. Self-Alignment Pretraining for Biomedical Entity Representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2021)*, pages 4228–4238, Online. Association for Computational Linguistics.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-Shot Entity Linking by Reading Entity Descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Daniel Loureiro and Alípio Mário Jorge. 2020. MedLinker: Medical Entity Linking with Neural Representations and Dictionary Matching. In *Proceedings of the 42nd European Conference on Information Retrieval (ECIR 2020)*, pages 230–237, Online. Springer International Publishing.

Sean MacAvaney and Craig Macdonald. 2022. A Python Interface to PISA! In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*, page 3339–3344, Madrid, Spain. Association for Computing Machinery.

Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, (OSIRRC@SIGIR 2019)*, pages 50–56, Paris, France.

Sunil Mohan and Donghui Li. 2019. MedMentions: A Large Biomedical Corpus Annotated with UMLS Concepts. In *Proceedings of the 2019 Conference on Automated Knowledge Base Construction (AKBC 2019)*, Amherst, MA, USA.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task (BioNLP 2019)*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

Mujeen Sung, Hwisang Jeon, Jinhyuk Lee, and Jaewoo Kang. 2020. Biomedical Entity Representations with Synonym Marginalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 3641–3650, Online. Association for Computational Linguistics.

Shikhar Vashishth, Denis Newman-Griffis, Rishabh Joshi, Ritam Dutt, and Carolyn P. Rosé. 2021. Improving broad-coverage medical entity linking with semantic type prediction and large-scale datasets. *Journal of Biomedical Informatics*, 121:103880.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *Preprint*, arXiv:2412.13663.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable Zero-shot Entity Linking with Dense Entity Retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 6397–6407, Online. Association for Computational Linguistics.

Zhenran Xu, Yulin Chen, and Baotian Hu. 2023. Improving Biomedical Entity Linking with Cross-Entity Interaction. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, pages 13869–13877, Washington, DC, USA. Association for the Advancement of Artificial Intelligence.

Sheng Zhang, Hao Cheng, Shikhar Vashishth, Cliff Wong, Jinfeng Xiao, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2022. Knowledge-Rich Self-Supervision for Biomedical Entity Linking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 868–880, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tiantian Zhu, Yang Qin, Ming Feng, Qingcai Chen, Baotian Hu, and Yang Xiang. 2024. BioPRO: Context-Infused Prompt Learning for Biomedical Entity Linking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:374–385.

A Statistical Significance

In our experiments in Section 6, we conduct statistical significance tests between every pair of models in our comparison. We only perform these statistical tests for the Accuracy@1 metric. As accuracy@1 values are binary (either 0 or 1), we apply a McNemar test with $p < 0.05$, where we pair the Acc@1 results for each mention in the test set. To account for the comparison of multiple models, we apply the Bonferroni correction. While Tables 3 and 4 summarize the statistical tests, we include in this appendix the complete statistical significance matrices.

These matrices are included in Figure 7 (MedMentions dataset), Figure 8 (NCBI Disease dataset), Figure 9 (NLM Chem) and Figure 10 (BC5CDR). On each matrix, a cell compares two algorithms: one indicated by the first row, and another indicated by the first column. A green cell represents a statistically significant difference between the two models ($p < 0.05$), whereas a white cell represents a non-significant difference ($p \geq 0.05$).

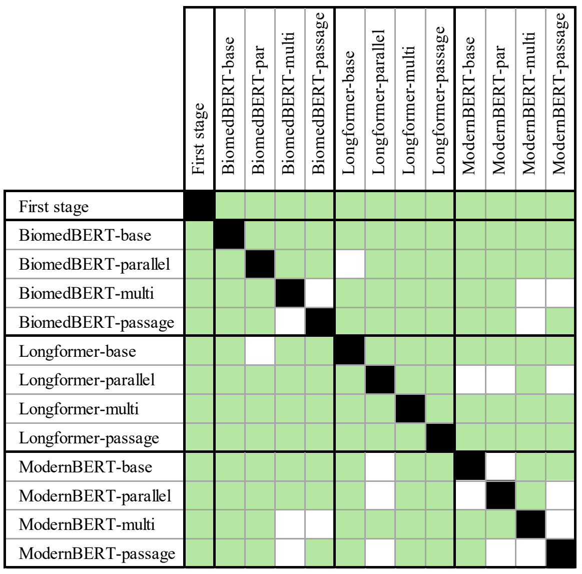


Figure 7: Acc@1 statistical significance (MedMentions)

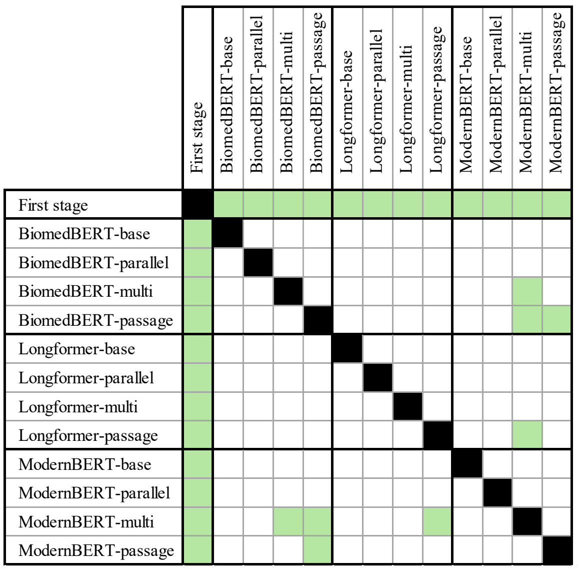


Figure 8: Acc@1 statistical significance (NCBI Disease)

B Input token length vs. training speed

We compare the average input token length and the training speed for the four datasets. Figure 11 (MedMentions), Figure 12 (NCBI Disease), Figure 13 (NLM Chem) and Figure 14 (BC5CDR) show the results. In all figures, x-axis shows the average input token length, and y-axis the training speed (in examples per second).

As we can observe, in general, in the smaller datasets, where the input token length is notably smaller than 4,096 (context window of the Longformer model), all models improve their training speed as we add more mention-entity pairs in their

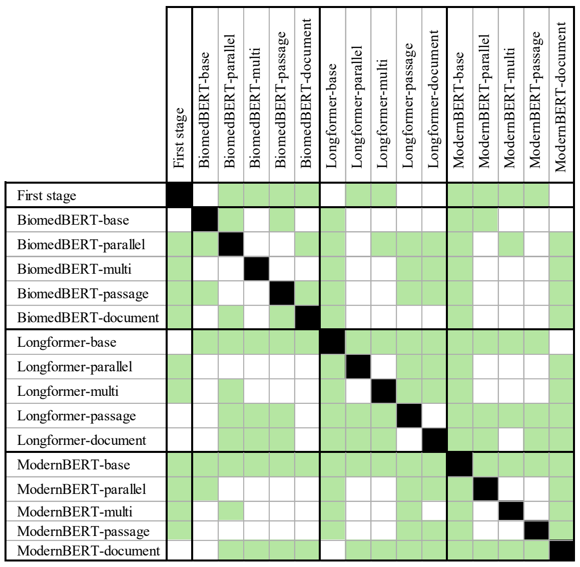


Figure 9: Acc@1 statistical significance (NLM Chem)

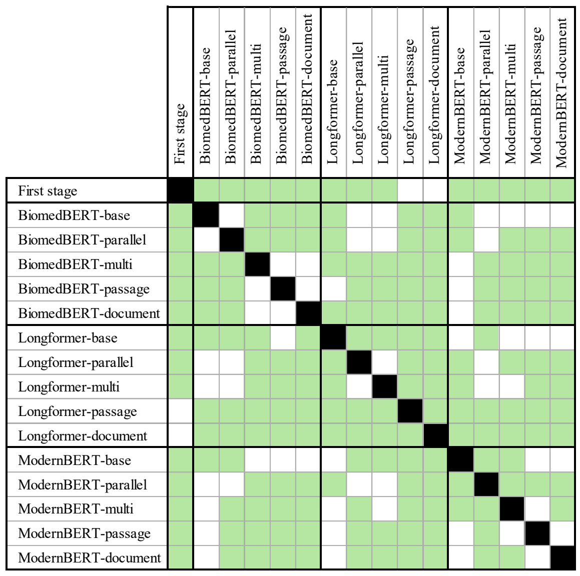


Figure 10: Acc@1 statistical significance (BC5CDR)

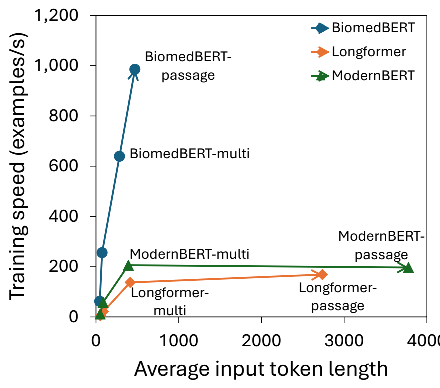


Figure 11: Input token length vs. training speed (Medmentions)

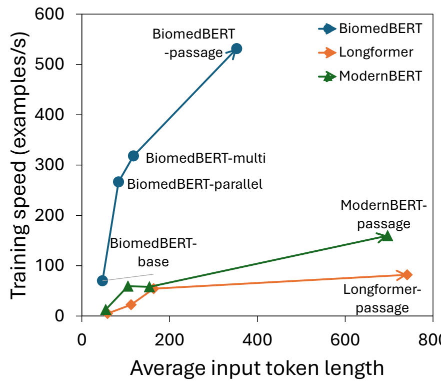


Figure 12: Input token length vs. training speed (NCBI Disease)

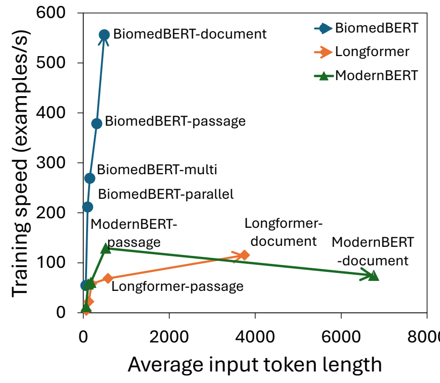


Figure 13: Input token length vs. training speed (NLM Chem)

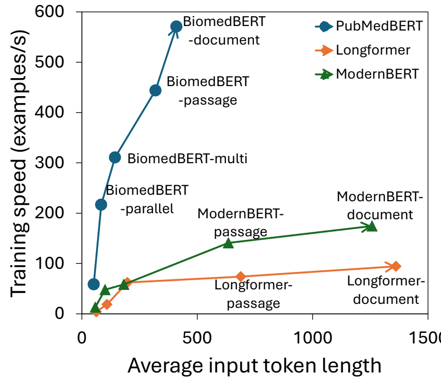


Figure 14: Input token length vs. training speed (BC5CDR)

input. However, in the bigger datasets (MedMentions and NLMChem), the ModernBERT model struggles with longer sequences (as observed in the training speeds of the ModernBERT-passage model in MedMentions and the ModernBERT-document model in NLM Chem).