Dual Cache for Long Document Neural Coreference Resolution

Qipeng Guo[1], Xiangkun Hu[1], Yue Zhang[2], Xipeng Qiu[3], Zheng Zhang[1]
[1]Amazon AWS AI, [2]School of Engineering, Westlake University
[3]School of Computer Science, Fudan University
`{gqipeng, xiangkhu, zhaz}@amazon.com`,
`zhangyue@westlake.edu.cn`, `xpqiu@fudan.edu.cn`

Abstract

Recent works show the effectiveness of cache-based neural coreference resolution models on long documents. These models incrementally process a long document from left to right and extract relations between mentions and entities in a cache, resulting in much lower memory and computation cost compared to computing all mentions in parallel. However, they do not handle cache misses when high-quality entities are purged from the cache, which causes wrong assignments and leads to prediction errors. We propose a new hybrid cache that integrates two eviction policies to capture global and local entities separately, and effectively reduces the aggregated cache misses up to half as before, while improving F1 score of coreference by $0.7 \sim 5.7$pt. As such, the hybrid policy can accelerate existing cache-based models and offer a new long document coreference resolution solution. Results show that our method outperforms existing methods on four benchmarks while saving up to 83% of inference time against non-cache-based models. Further, we achieve a new state-of-the-art on a long document coreference benchmark, LitBank.

1 Introduction

Coreference Resolution (CR) is fundamental in document-level Natural Language Processing. Its goal is to identify mentions belonging to the same entity. These mentions often scatter throughout the document, but they can be linked together through coreference resolution. CR is a building block for common sense understanding (Levesque et al., 2012; Sakaguchi et al., 2021; Balahur et al., 2011; Liu et al., 2021a), reading comprehension (Dasigi et al., 2019; Storks et al., 2019), information extraction (Yao et al., 2019; Ji and Grishman, 2008; Lu and Ng, 2018), and text summarization (Liu et al., 2021b; Azzam et al., 1999; Xu et al., 2020).

Conventional CR models enumerate every pair of mentions in a document in parallel, so the computation and memory cost is quadratic to the number of mentions in a document (Xia et al., 2020), where mention is a pronoun, a noun phrase, or a text span that can be referred. This quadratic overhead poses a challenge for long-doc CR. Recent work (Xia et al., 2020; Toshniwal et al., 2020, 2021) proposes cache-based CR models which scan mentions in a document from left to right, storing resolved entities in a cache and determining whether to assign a new mention to an entity in the cache or push it to the cache as a new entity. When the cache is full, it will evict entities according to a certain eviction policy, such as LRU (Least Recently Used). We denote the cache with LRU eviction policy as a local cache (L-cache) because it keeps the more recent entities. Since the small cache size bounds the number of potential coreference relations that the model probes for each mention, it reduces the computation and memory cost. Specifically, the cost reduces to $O(|C||D|)$ from $O(|D|^2)$, where the cache size $|C| \ll |D|$, $|C|$ and $|D|$ are cache size and document length, respectively.

However, multiple topics are common when doc-

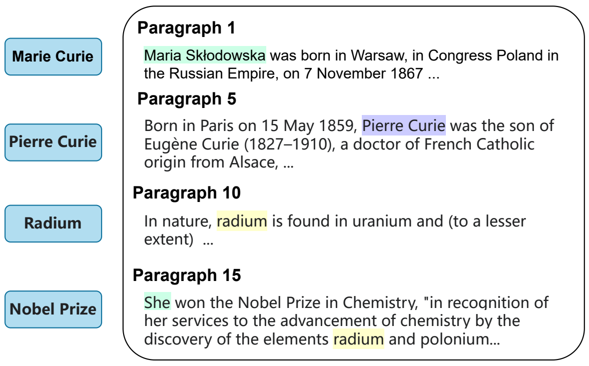

Marie Curie
Pierre Curie
Radium
Nobel Prize
Paragraph 1
Maria Skłodowska was born in Warsaw, in Congress Poland in the Russian Empire, on 7 November 1867 ...
Paragraph 5
Born in Paris on 15 May 1859, Pierre Curie was the son of Eugène Curie (1827–1910), a doctor of French Catholic origin from Alsace, ...
Paragraph 10
In nature, radium is found in uranium and (to a lesser extent) ...
Paragraph 15
She won the Nobel Prize in Chemistry, "in recognition of her services to the advancement of chemistry by the discovery of the elements radium and polonium...

Figure 1: For a long document, the topic switching often happens, such as the discussion might shift from "Marie Curie" to "Pierre Curie," followed by "Radium," and then "Nobel Prize." These shifts in topic inevitably lead to certain entities being temporarily excluded from the ongoing discussion but reintroduced after a series of topic changes.

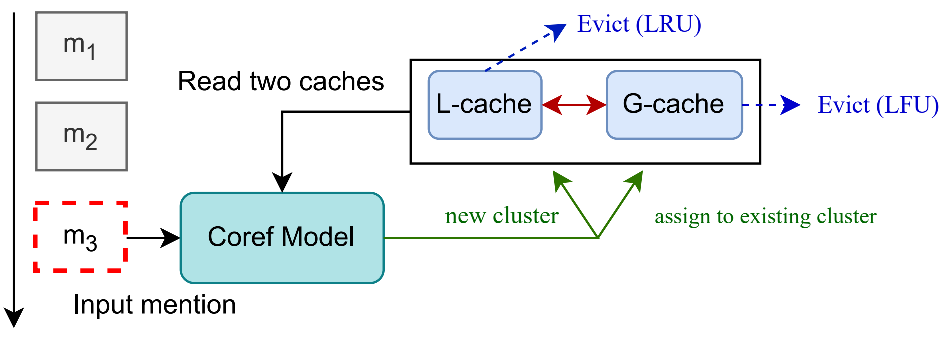


Figure 2: Overview of our method. The model scans the mentions $m_1, m_2, \cdots$ in a document from left to right. There are two caches whose eviction strategy follows recent usage and frequency, respectively. For each mention, the model decides whether to push it into caches as a new entity cluster or assign it to an existing entity cluster. If the frequency of an entity is higher than any entity in the G-cache, we will push it into the G-cache. Otherwise, we will push it into the L-cache. Note that if an entity in the L-cache has a higher frequency than any entity in the G-cache, we switch them as shown with the red edge in the figure. If a cache is full, it evicts the entity according to the cache policy.

ument is long and the focus of narrative may switch (Figure 1). This phenomenon impacts the performance of LRU policy due to the repeated alternation of topics, causing an entity to be mentioned after a long span of text, increasing its chance of eviction from the cache. When processing a mention that requires the evicted entity, it results in a cache miss. Empirically, we find 11.1% mentions encounter cache misses when adopting LRU policy on real data (see Section 5.1).

Our analysis of cache miss reveals that the pattern of entity usages follows the Pareto principle (Koch, 2011) (or 80-20 rule). In other words, a few high frequency entities are used globally (i.e., key concepts for each topic) and account for most of the cache misses. In LitBank (Bamman et al., 2019), 6% of the entities account for 97% of the cache miss. Another example that fits this observation is that the 20 main characters are mentioned 1,722 times in Animal Farm (Orwell, 2021), a book of 30,000 words (see Section 5.5).

Inspired by this insight, we propose a dual cache to address different patterns of word usage. One cache stays unchanged as before, i.e., an L-cache using LRU policy. Another cache is devoted to global and high frequent entities evicted from L-cache, called G-cache. Intuitively, G-cache adopts the classical cache policy LFU (Least Frequently Used). We denote the proposed hybrid cache as Dual cache. The idea is to promote a division of labor in order to make more effective use of cache: L-cache deals with local, clustered mentions, typically in one topic, whereas G-cache targets global entities within a longer range, typically across topics. A diagram of a model with Dual cache is shown in Figure 2.

We conduct extensive experiments on coreference resolution benchmarks, including OntoNotes (Pradhan et al., 2012), LongtoNotes (Shridhar et al., 2022), LitBank (Bamman et al., 2019), and WikiCoref (Ghaddar and Langlais, 2016). Results show that a CR model with our Dual cache outperforms the prior work (Toshniwal et al., 2020; Xia et al., 2020) which use unbounded cache while saving 56% of the inference time. The model achieves a new state-of-the-art on LitBank (Bamman et al., 2019).

The results shown in Section 4 demonstrate the effectiveness of the Dual cache on long-doc CR benchmarks. To further explore the capability of our Dual cache on long documents, we annotate a book with 30,000 words, our method outperforms the baseline by offering 10 points improvement of F1 while saving 70% of the inference time. The data and code are available `https://github.com/QipengGuo/dual-cache-coref`.

2 Related Work

Conventional CR methods (Joshi et al., 2019; Xu and Choi, 2020; Jiang and Cohn, 2021) are designed for short documents with length less than 1,000 words. They typically encode the input document and propose candidate mentions with a pre-trained model. Their model enumerates all pairs of mentions in parallel to identify coreference relations, resulting in memory and computation cost quadratic to document length. These models also require a post-processing to gather connected mentions as entities and need to compute a large tensor of $\mathcal{O}(n \times n \times d)$, where n is the number of mentions, and d is the dimension of hidden states in the relation classifier. This is often impractical for long documents with thousands of mentions. Another direction is to reformulate CR, Wu et al. (2020) treats CR as a question answering problem and achieves higher performance on some benchmarks but costs more computation since the model needs to do question answering repeatedly.

Cache-based Neural Coreference Resolution Since directly applying conventional CR on long documents suffers from heavy computation and memory cost, recent works (Xia et al., 2020; Toshniwal et al., 2020, 2021) proposed cache-based CR models. Probing entities in the cache instead of the whole document largely saves the computation and memory cost but also brings errors when encountering a cache miss. Thus, the eviction policy plays an important role. Xia et al. (2020) follow the LRU principle, and Toshniwal et al. (2020) discuss more eviction policies, such as a variant of LRU and a learnable scorer to rank elements. However, these works lack the ability to capture the topic switching phenomena described in Figure 1.

Datasets for Long document Coreference Existing CR datasets are typically on short documents, such as the most commonly used corpus, OntoNotes (Pradhan et al., 2012), whose average document length is 487 words. However, as Bamman et al. (2019) have shown, text in scientific papers and literature, which can be much longer than OntoNotes articles, exhibit different usages of coreference relations. Note that the original document collected for OntoNotes is much longer, but they split documents to lower the annotation cost, and a recent work LongtoNotes(Shridhar et al., 2022) attempts to recover the long document. Besides, recent work proposes benchmarks for long documents, such as LitBank (Bamman et al., 2019) for literature articles, ACL-Coref (Schäfer et al., 2012) for scholarly papers, and Wiki-Coref (Ghaddar and Langlais, 2016) for documents in Wikipedia. These corpora not only increase document length against traditional coreference benchmarks but also largely increase the spread of entities (the distance between the first and the last mention of an entity , which is defined in Toshniwal et al. (2020)), which requires the model to memorize longer history. Text books, story books, and professional books capture much of the human knowledge, but book-scale CR has not caught attention, and we made the first attempt to address this issue. We annotate a 30,000-word book, Animal Farm, taking the 20 characters as entities, and annotate the 1,722 mentions of them throughout the book.

3 Method

A common workflow of neural CR models is to detect candidate mentions, encode them into vector representations, and identify coreference relations between each mention and past entities by feeding their representations to an MLP classifier. We denote candidate mentions as $\{m_1, m_2, \cdots, m_n\}$, and an entity is defined as a set of mentions, $e_k = \{m_1^k, m_2^k, \cdots\}$. We use a special operator $H(\cdot)$ to represent vector representations, for example, $H(m_1)$ is the representation of the mention m_1. The key component of our method is a dual cache which contains an L-cache $C_L = \{e_1, \cdots, e_{N_L}\}$ and a G-cache $C_G = \{e_1, \cdots, e_{N_G}\}$, where e is an entity in the cache, N_L and N_G are the sizes of the two caches respectively. Since we focus on cache design, we adopt Longformer (Beltagy et al., 2020) as the document encoder and mention detection model (an MLP scorer) from Toshniwal et al. (2021) which are described in Toshniwal et al. (2020).

3.1 Dual Cache

When an entity is brought to use, it tends to be reused in a local context intensively, exhibiting clustering behavior that an LRU can effectively capture. As mentioned above, topic switching may cause some entities to be intensively used in certain paragraphs scattered in the document but completely not used in others. Thus, these entities are sensitive to be purged by an LRU cache due to a long absence, but their defining characteristic is its high frequency, easily captured by an LFU policy. As such, neither policy can handle both patterns, and the integration is an intuitive way to cover both cases.

We adopt a divide-and-conquer approach to use two caches and adopt different eviction policies. L-cache captures the locally active terms and follows the LRU policy. G-cache focuses on the global terms and follows the LFU policy. When a new mention comes, we first ask the model to classify whether it is a new entity or is is assigned to an existing entity within the cache. Next, we will check the frequency of this new or updated entity, and we will move it to the G-cache if its frequency is higher than any entity in the G-cache and trigger an eviction of the G-cache when it is full. If the entity does not enter the G-cache, we will move it to the L-cache and trigger an eviction of the L-cache when it is full. L-cache will evict the least recently used entity and G-cache will evict the least frequently used entity.

At the beginning of processing a document, both L-cache and G-cache are empty, and we will fill the G-cache first since the frequency of an empty slot equals zero and the frequency of a newly entered entity is at least one.

3.2 Coreference Resolution with a Dual Cache

We follow Toshniwal et al. (2021) to design an incremental neural coreference model, which scans mentions one by one and computes pair-wise scores for entities in the cache plus a placeholder vector indicating new entity (noted as $\mathcal{N}$). As its name implies, if this placeholder achieves the highest prediction score, the model will create a new entity and push it to the cache. Otherwise, the model assigns the input mention to the entity with the highest score and updates its representation. Note that the cache structure is transparent to the coreference resolution model, and the model will access all entities in the Dual cache.

The score between a mention m, an entity e in the Dual cache $C = \{C_L, C_G\}$, and a placeholder vector $\mathcal{N}$ can be computed as,

$$s(m, \mathcal{N}) = f_c(\mathbf{H}(m), \mathcal{N}), \tag{1}$$

$$s(m, e) = f_c(\mathbf{H}(m), \mathbf{H}(e)), \qquad e \in C \tag{2}$$

where $\mathbf{H}(m)$, $\mathbf{H}(e)$ are representations of a mention and an entity output by a pre-trained language model, respectively. f_c means an MLP classifier.

For the dataset that requires the model to distinguish singletons, we record the $s(m, \mathcal{N})$ as a measure. For an entity only contains one mention after processing the whole document, if the recorded score exceeds the threshold (set to 0 in this work), the model determines it as a singleton. Otherwise, it will not be considered as entity and dropped from the prediction results.

When the model assigns a new mention to an existing entity, the entity representation gets updated by a gated sum aggregator.

$$\text{Update}(e, m) = \alpha \mathbf{H}(m) + (1 - \alpha)\mathbf{H}(e), \tag{3}$$

$$\text{where} \quad \alpha = f_g(\mathbf{H}(e), \mathbf{H}(m)), \tag{4}$$

where α is the aggregation coefficient and f_g is an MLP module with sigmoid function. For a new created entity, its representation is set with the first mention's representation.

3.3 Learning and Inference

There are three learnable components in our method, a mention-level representation extractor, which consists of a pre-trained language model and an MLP classifier and is the same as Toshniwal et al. (2021), a pair-wise scoring module eq. (2) to classify coreference relations, and an update function eq. (3) used to update clusters' representation when assigning new mentions to an entity. Since we only introduce a new cache policy, we can reuse other cache-based models' parameters like Xia et al. (2020) and Toshniwal et al. (2021) to perform coreference resolution without further training.

We can also retrain or finetune the model with the Dual cache for long documents, which has two potential advantages: 1) the Dual cache has a higher cache hit ratio so that it contains more ground-truth entities, which means the model can see more positive relations during training; 2) it avoids feature space shift since we get the representation of entities by merging mentions sequentially. Different cache polices lead to different merging orders of mentions so that affect the entities' representation.

We follow Toshniwal et al. (2021) to adopt Cross Entropy as the loss function and take the ground-truth of coreference relations and mention boundaries as labels.

4 Experiments

We conduct experiments on four CR benchmarks and compare with prior cache-based CR methods.

4.1 Datasets

The four datasets for our experiments are: OntoNotes and its extension LongtoNotes for longer documents; WikiCoref with only testing documents; LitBank which provides an official split of ten-fold cross-validation and we follow prior work (Toshniwal et al., 2021) to report the result of the first fold in this work. The statistics of the datasets are listed in Table 1.

Dataset	Train	Val	Test	Avg. Length
OntoNotes	2,802	343	348	487
LongtoNotes	1,959	234	222	674
WikiCoref	-	-	30	1,996
LitBank	80	10	10	2,105
Animal Farm	-	-	1	37,000

Table 1: Statistics of the datasets for experiments, including the numbers of documents in each split and the average number of tokens of documents.

4.2 Baselines

Xia et al. (2020) proposed a simple cache-based CR model built on SpanBERT (Joshi et al., 2020) with an L-cache. The model architecture is inherited from a strong baseline of conventional CR model (Joshi et al., 2019) and is finetuned based on their released model. They freeze SpanBERT during finetuning and only report results on OntoNotes instead of benchmarks with longer documents.

Toshniwal et al. (2021) [1] provided more results of cache-based CR models for long documents, including results on OntoNotes, LitBank, and WikiCoref. They replace the backbone encoder with Longformer (Beltagy et al., 2020) for better performance and update all parameters during finetuning. We follow their training scripts and hyperparameters [2] to train our model.

Thirukovalluru et al. (2021) largely accelerates the model by changing span-level CR to token-level CR, which considers the relations between token pairs first, and then maps them as relations between mentions and entities. However, looking at the tokens in two mentions is not enough to identify a coreference relation, especially for long mentions. Their code [3] was not released when writing this paper. Thus, we only compare the F1 score with them. They report the result on a book of 2M tokens. Since their predictions are not released, we plan to compare with them on book-scale document in the future.

4.3 Setup

We initialize our model with the parameters released by Toshniwal et al. (2021), including a document encoder, a mention proposer, and a CR model. We also use a unified mention detection strategy

[1]We do not compare with their previous work (Toshniwal et al., 2020) since the experiment setting is slight difference, and the compared one is an upgrade version in terms of both performance and methodology.

[2]`github.com/shtoshni/fast-coref`

[3]`github.com/raghavlite/Scalable-Coreference`

Method	Cache Size	Test set F1	S-subset F1
Toshniwal et al. (2021)	Unbound	77.8	76.4
Toshniwal et al. (2021)	50	72.6	71.0
Dual cache	25+25	**76.2**	**74.8**
Toshniwal et al. (2021)	200	77.2	75.4
Dual cache	100+100	**77.7**	**76.3**
Toshniwal et al. (2021)	500	77.7	76.0
Dual cache	250+250	**77.9**	**76.3**

Table 2: Results on LongtoNotes. "Test set" refers to the normal test set and "S-subset" refers to the long-doc subset, LongtoNotes$_S$. Due to the limited space, we report the variance of three runs in the Appendix C.

and keep the top $0.4|D|$ candidate mentions as Toshniwal et al. (2021) did for a fair comparison except the one on Animal Farm, where $|D|$ is the document length. We focus on CR model and report its computation cost and inference time, as the document encoder and mention proposer enable plug-and-play. All experiments are conducted on NVIDIA T4 GPU (16GB). We report MAC (Multiply Accumulate operations) to measure the computation cost. [4]

4.4 Main Results

Table 3 compares our method and prior method on three benchmarks. We denote the size of the Dual cache as “A+B", meaning there are A slots in the L-cache and B slots in the G-cache. We report the F1 score and inference time as measurements of performance and efficiency, respectively. We list more metrics like MUC (Vilain et al., 1995), B-CUBED (Bagga and Baldwin, 1998), and CEAFE (Luo, 2005) in the Appendix C.

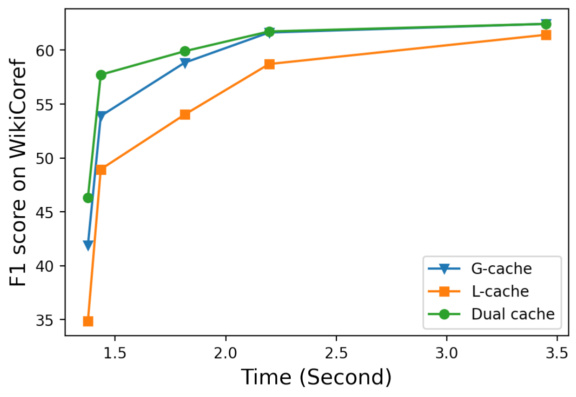
Besides the overall performance, we compare with the strongest baseline (Toshniwal et al., 2021) under the setting of different cache sizes. Results demonstrate that our method consistently outperforms previous approaches, albeit with a slightly higher time consumption for a given cache size. Furthermore, our method achieves comparable performance in less inference time. In particular, our Dual cache with 50 slots outperforms the previous SOTA approach that utilizes an unbounded cache on LitBank, while only requiring 48% of the inference time.

Additionally, we provide results on LongtoNotes, a recent extension of OntoNotes that combines pas-

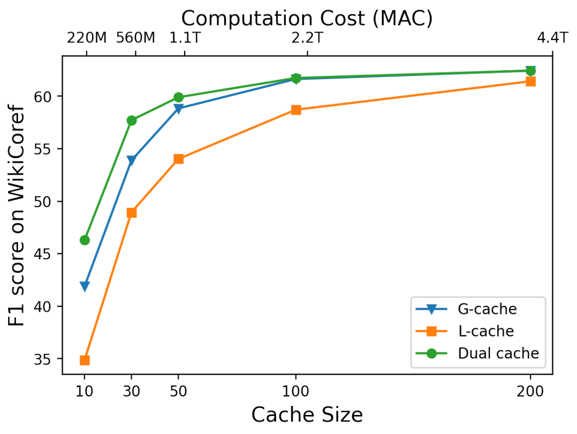
[4]The script for calculating MAC is adapted from `https://github.com/Lyken17/pytorch-OpCounter/`

Method	Cache Size	LitBank F1	LitBank Time	OntoNotes F1	OntoNotes Time	WikiCoref F1	WikiCoref Time	Avg F1	Avg Time
Thirukovalluru et al. (2021)†	Unbound	75.9	-	78.0	-	-	-	-	-
Xia et al. (2020)	Unbound	76.7	5.72s	79.6	0.98s	58.7	12.35s	71.6	6.35s
Toshniwal et al. (2021)	Unbound	78.6	5.17s	80.6	0.66s	63.5	13.72s	74.2	6.52s
Toshniwal et al. (2021)	50	72.9	2.14s	75.1	0.40s	53.0	2.10s	67.0	1.55s
Dual cache	25+25	**78.8** ± **0.33**	**1.95s**	**79.6** ± **0.21**	**0.41s**	**59.9**	**2.20s**	**72.7** ± **0.18**	**1.52s**
Toshniwal et al. (2021)	200	78.2	3.32s	79.8	0.53s	61.4	3.96s	73.1	2.60s
Dual cache	100+100	**79.3** ± **0.32**	**3.53s**	**81.0** ± **0.18**	**0.61s**	**62.4**	**4.17s**	**74.3** ± **0.17**	**2.77s**
Toshniwal et al. (2021)	500	78.5	4.77s	80.0	0.55s	62.8	6.63s	73.8	3.98s
Dual cache	250+250	**79.5** ± **0.37**	**5.09s**	**81.1** ± **0.19**	**0.63s**	**63.0**	**7.15s**	**74.5** ± **0.18**	**4.29s**

Table 3: Results on three benchmarks. "Cache size" means the maximum number of entities in the cache, and "Unbound" means the cache stores all the entities. "F1" is the F1 score, and "Time" is the inference time. Since WikiCoref only has the test set, we report the result of integrating Toshniwal et al. (2021) with our Dual cache. We report the standard deviation of three runs and omit the inference time variance since it is less than 0.01s. † means the work that does not publicly release the code, so we only report the F1 score here.



(a) Inference Time vs. F1 score



(b) Cache size and Computation Cost vs. F1 score

Figure 3: Efficiency vs. Performance on WikiCoref, where computation cost and inference time are reported on the test set. The two regions of Dual caches are half-and-half allocated.

sages into longer documents. Table 2 exhibits the same trend, where our Dual cache outperforms the baselines on both the complete test set and a subset of long documents. Notably, the performance gain is more significant in the long document subset, highlighting the effectiveness of the Dual cache structure in processing such documents.

To better illustrate the effectiveness and efficiency, we report more results on WikiCoref in Figure 3. In this setting, we adopt a CR model trained with unbounded memory, so it does not have a preference of the cache structure. The curves show that the Dual cache always has the highest performance/cost ratio among different methods and outperforms both the L-cache and the G-cache either using a fixed cache size or consuming a fixed amount of time, demonstrating the benefit of integrating the L-cache and G-cache.

5 Analysis

5.1 Cache Miss Ratio

By comparing the performance gain when using different cache sizes in Table 3 and Table 2, we find that the improvement of the Dual cache is more significant for a small cache, such as the performance gain for 50 slots is 5.7pt F1 and the gain for 200 slots is 1.2pt F1. The reason is that the cache miss ratio decreases rapidly as the cache gets larger, so the absolute improvement of our method is also getting smaller.

In this section, we quantitatively discuss how the Dual cache reduces the cache miss ratio. We adopt an off-the-shelf mention detection model (Toshniwal et al., 2021) to detect mentions and use ground truth to replace the relation classifier to get rid of the model difference, except the cache structure.

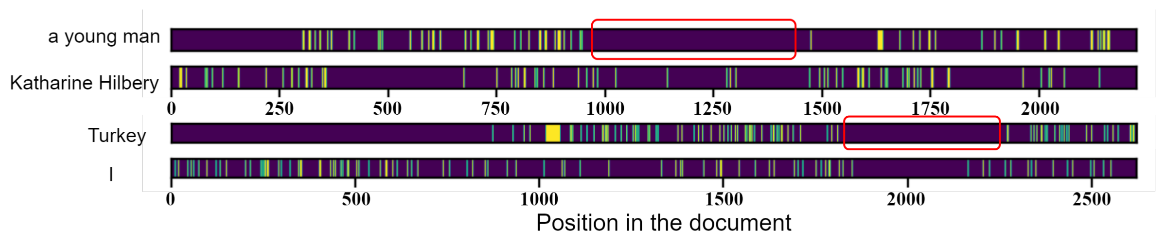


Figure 4: Heatmap of entities' occurrences in a document. Each entity is labeled based on its first appearance in the document. Regions where an L-cache would evict an entity due to insufficient cache size (less than 300) are highlighted with red boxes. These regions indicate potential cache misses. In contrast, the Dual cache can avoid such errors.

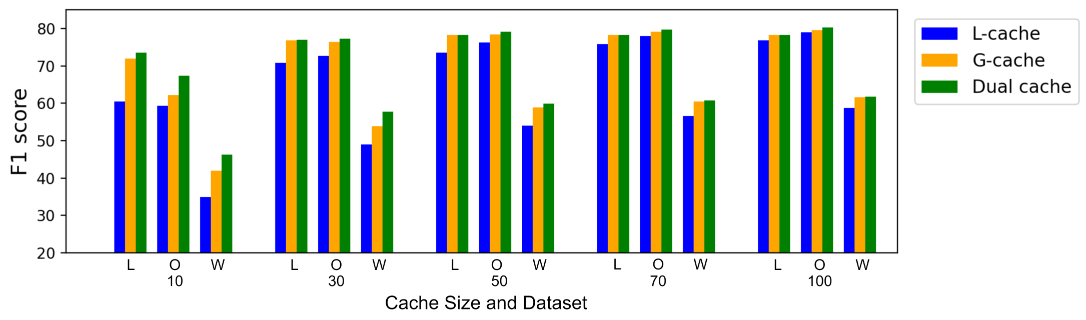


Figure 5: Results on different benchmarks with different cache sizes. We use "L" for LitBank, "O" for OntoNotes, and "W" for WikiCoref. All models' parameters used here are loaded from the baseline without finetuning.

Model	Cache Size 10	30	50	70	100
	OntoNotes				
L-cache	7.4%	2.8%	1.5%	0.98%	0.54%
G-cache	6.4%	3.7%	2.5%	1.8%	1.1%
Dual cache	**5.0%**	**1.5%**	**0.82%**	**0.51%**	**0.27%**
	LitBank				
L-cache	7.2%	3.1%	1.9%	1.4%	0.98%
G-cache	3.9%	2.3%	2.0%	1.9%	1.5%
Dual cache	**3.5%**	**1.2%**	**0.69%**	**0.48%**	**0.33%**
	WikiCoref				
L-cache	11.1%	6.3%	4.8%	4.0%	3.2%
G-cache	8.4%	5.8%	4.8%	4.2%	3.5%
Dual cache	**7.8%**	**4.1%**	**3.0%**	**2.4%**	**1.8%**

Table 4: Cache miss ratio on three benchmarks.

Table 4 gives the cache miss ratio for different caches on three benchmarks. Results show that the Dual cache reduces the chance of cache miss significantly, especially in the case of 10 slots and 30 slots. A cache miss will lead to an inevitable prediction error since the groundtruth does not lie in the solution space, which means the model only considers assigning the mention to clusters within the cache, but the groundtruth cluster is not in the cache. Figure 5 illustrates the impact of cache miss ratio, indicating a lower cache miss ratio and suggesting a higher performance.

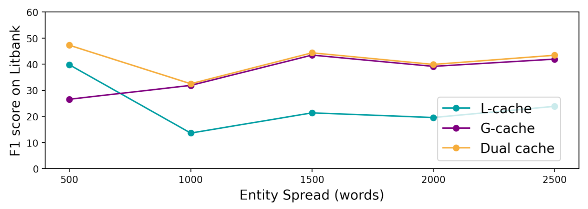


Figure 6: Entity spread vs. F1 score on WikiCoref. We report models with a 50 slots L-cache, a 50 slots G-cache, and a 25+25 slots Dual cache.

In addition, we provide case studies of four entities in Figure 4, which shows the heatmap of an entity's occurrences. The color reveals the density of occurrences. A light color means the entity is frequently used there, and a dark color means no usage. We highlight regions with long periods of an entity's absence by red boxes. The region length can reach 500 words, meaning the L-cache will evict it unless the cache has more than 500 slots. The second case ("Katharine Hilbery") contains multiple relatively long periods (>100 words) of absence. If the cache size is less than 100, it will encounter the cache miss 5 times and cause errors.

5.2 Impact of Entity Spread

As we have shown in Figure 4, an entity may spread in a long range in a document, and this type of en-

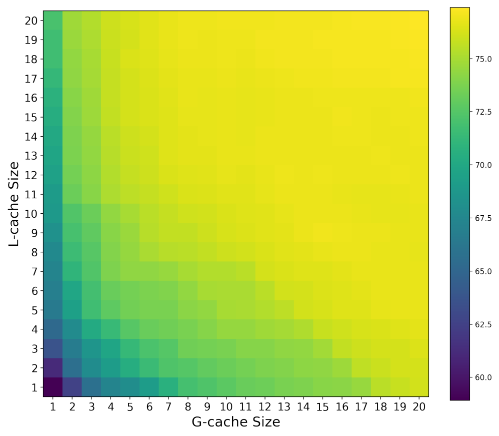


Figure 7: F1 score on LitBank for different cache sizes. The X-axis represents the size of the G-cache, and the Y-axis represents the size of the L-cache.

tities is challenging for cache mechanisms. One basic measurement is the Entity Spread, which describes how an entity scatters throughout a document. A large spread means the entity spans a large text area, which is likely to be missed by an L-cache if the L-cache is not large enough and the entity is absent for a while.

Figure 6 reports models' results on different entity spread scales. We show that the Dual cache helps the model resolve entities with a large spread, resulting in nearly a 30 F1 gain. The reason is that an entity with a large spread is often frequently used, which can be captured by G-cache easily. We can also see that L-cache is good at entities with small spreads while G-cache is good at entities with large spreads, and the results also demonstrate that neither of them could solve the coreference alone. The dual cache achieves better results by taking advantage of both L-cache and G-cache.

5.3 Impact of Cache Size

Cache size is an essential factor for a cache design, and there are two caches in our Dual Cache structure, so we study the influence of total cache size and the allocation of L-cache and G-cache and discuss them in this section.

We test our method with the cache size varying from 10 to 100. We report the results of the models without finetuning to avoid the potential noise of introducing new parameters and training processes. Figure 5 shows results on three benchmarks, and we can see the Dual cache outperforms both the L-cache and the G-cache, especially for a small cache.

Since the Dual cache consists of two caches, it is valuable to discuss how their allocation affects the performance. We provide a heatmap in Figure 7, where the X-axis is the size of G-cache and the Y-axis is the size of L-cache. We enumerate all their combinations (200 settings) when varying the size from 1 to 20. Results show the G-cache is more important when the total size is less than 20 (the lower triangle). The allocation of two caches becomes less matter when the total size exceeds 20 (the upper triangle).

	Cache Size				
Model	10	30	50	70	100
	OntoNotes				
L-first	5.1%	1.5%	0.80%	0.53%	0.28%
G-first	5.0%	1.5%	0.82%	0.51%	0.27%
	LitBank				
L-first	3.4%	1.2%	0.71%	0.50%	0.34%
G-first	3.5%	1.2%	0.69%	0.48%	0.33%
	WikiCoref				
L-first	7.8%	4.0%	2.9%	2.4%	1.9%
G-first	7.8%	4.1%	3.0%	2.4%	1.8%

Table 5: Cache miss ratio of Dual cache using different priorities on OntoNotes, LitBank, and WikiCoref.

Method	F1	Time	Memory
Toshniwal et al. (2021)	25.8	1284s	5.2 GB
Dual cache	36.3	375s	4.5 GB

Table 6: Results on Animal Farm.

5.4 Priority of two caches

The positions of the L-cache and the G-cache are not equivalent in our Dual cache. What we have described in this work is actually a G-first Dual cache, which means the priority of the G-cache is higher than the L-cache. Particularly, it means we will push an entity to the G-cache when both the G-cache and L-cache has empty slots. We also tried the L-first version and report it in Table 5. The results show that the difference between G-first and L-first is around 0.1%, which is negligible.

5.5 Results on Book-Scale CR

The proposed Dual cache structure is for long-doc CR. However, the maximum length of documents in the four benchmarks is fewer than 10,000 words. To test our approach on a more challenging scenario, we annotate a book of approximately 30,000 words, Animal Farm. We choose the 20 characters in this book as entities and annotate all their 1,722 mentions across the book. Appendix A describes the details of the book annotation. We compare our model and the baseline trained on LitBank since its documents are also from books. We tokenize the book into 37,000 tokens and split them into 11 segments so that we can feed them into the document encoder. We set the cache size to 500 for the baseline and 250+250 for the Dual cache for a fair comparison.

Our model can parse the entire book in 10 minutes when selecting $0.4|D|$ top-scored mentions, but the baseline can not finish the inference process in an hour. Thus, we reduce the number of mentions to $0.3|D|$ without changing other settings. We report F1 score, inference time, and inference memory in Table 6. The results reveal that the model with a Dual cache significantly outperforms the baseline in both efficiency and effectiveness on a book-scale document.

6 Conclusion

We proposed a new cache structure, Dual cache, to tackle long document coreference resolution. It contains two caches governed by different eviction polices. The local cache follows LRU policy to deal with local, clustered mentions, whereas the global cache follows LFU policy to target global entities within a long span. Empirical results show that the Dual cache lowers the cache miss up to half as before and improves the F1 score consistently on four benchmarks by offering an average gain of 5.7 F1 score for small caches while taking the same inference time. We also achieve a new state-of-the-art, 79.5 F1 score, on LitBank.

Limitations

This work is motivated by the intuition that a mention is more likely to refer to an entity that occurs shortly earlier and refers to a high frequency entity but has not recently used. For the latter pattern, we show examples of topic switching to explain why these phenomena happened, but we have not found a rigorous linguistic explanation to support this finding. We provide empirical results on four benchmarks plus a book. However, the scarcity of long-doc CR benchmarks hinders us from verifying on a larger scale.

Our major contribution is a new cache design, but we also find the cache design becomes less matter when using a huge cache. NVIDIA A100 has 80G memory, which means it can handle a document of 100,000 words with a conventional CR model. As the GPU becomes larger and cheaper, the importance of studying cache design is weaker.

Ethics Statement

We comply with the ACL Ethics Policy. Coreference resolution is fundamental in NLP, which often serves as a component of other NLP applications. Since it does not directly interact with users, there are no additional ethics concerns. All the data used in this paper is public. We confirm that the scientific artifacts used in this paper comply with their license and intended use. We list the licenses in Table 7.

References

Saliha Azzam, Kevin Humphreys, and Robert J. Gaizauskas. 1999. Using coreference chains for text summarization. In *COREF@ACL*. Association for Computational Linguistics.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

Alexandra Balahur, Jesús M. Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *WASSA@ACL*, pages 53–60. Association for Computational Linguistics.

David Bamman, Sejal Popat, and Sheng Shen. 2019. An annotated dataset of literary entities. In *NAACL-HLT (1)*, pages 2138–2144. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *EMNLP/IJCNLP (1)*, pages 5924–5931. Association for Computational Linguistics.

Abbas Ghaddar and Philippe Langlais. 2016. Wikicoref: An english coreference-annotated corpus of wikipedia articles. In *LREC*. European Language Resources Association (ELRA).

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*, pages 254–262. The Association for Computer Linguistics.

Fan Jiang and Trevor Cohn. 2021. Incorporating syntax and semantics in coreference resolution with heterogeneous graph attention network. In *NAACL-HLT*, pages 1584–1591. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel S. Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *EMNLP/IJCNLP (1)*, pages 5802–5807. Association for Computational Linguistics.

Richard Koch. 2011. *The 80/20 Principle: The Secret of Achieving More with Less: Updated 20th anniversary edition of the productivity and business classic*. Hachette UK.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *KR*. AAAI Press.

Chunhua Liu, Trevor Cohn, and Lea Frermann. 2021a. Commonsense knowledge in word associations and conceptnet. In *CoNLL*, pages 481–495. Association for Computational Linguistics.

Zhengyuan Liu, Ke Shi, and Nancy F. Chen. 2021b. Coreference-aware dialogue summarization. In *SIGDIAL*, pages 509–519. Association for Computational Linguistics.

Jing Lu and Vincent Ng. 2018. Event coreference resolution: A survey of two decades of research. In *IJCAI*, pages 5479–5486. ijcai.org.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT/EMNLP*, pages 25–32. The Association for Computational Linguistics.

George Orwell. 2021. *Animal farm*. Oxford University Press.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*, pages 1–40. ACL.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Ulrich Schäfer, Christian Spurk, and Jörg Steffen. 2012. A fully coreference-annotated corpus of scholarly papers from the ACL anthology. In *COLING (Posters)*, pages 1059–1070. Indian Institute of Technology Bombay.

Kumar Shridhar, Nicholas Monath, Raghuveer Thirukovalluru, Alessandro Stolfo, Manzil Zaheer, Andrew McCallum, and Mrinmaya Sachan. 2022. Longtonotes: Ontonotes with longer coreference chains. *CoRR*, abs/2210.03650.

Shane Storks, Qiaozi Gao, and Joyce Y. Chai. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *CoRR*, abs/1904.01172.

Raghuveer Thirukovalluru, Nicholas Monath, Kumar Shridhar, Manzil Zaheer, Mrinmaya Sachan, and Andrew McCallum. 2021. Scaling within document coreference to long texts. In *ACL/IJCNLP (Findings)*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3921–3931. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020. Learning to ignore: Long document coreference with bounded memory neural networks. In *EMNLP (1)*, pages 8519–8526. Association for Computational Linguistics.

Shubham Toshniwal, Patrick Xia, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2021. On generalization in coreference resolution. *CoRR*, abs/2109.09667.

Marc B. Vilain, John D. Burger, John S. Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC*, pages 45–52. ACL.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. Corefqa: Coreference resolution as query-based span prediction. In *ACL*, pages 6953–6963. Association for Computational Linguistics.

Patrick Xia, João Sedoc, and Benjamin Van Durme. 2020. Incremental neural coreference resolution in constant memory. In *EMNLP (1)*, pages 8617–8624. Association for Computational Linguistics.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *ACL*, pages 5021–5031. Association for Computational Linguistics.

Liyan Xu and Jinho D. Choi. 2020. Revealing the myth of higher-order inference in coreference resolution. In *EMNLP (1)*, pages 8527–8533. Association for Computational Linguistics.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *ACL (1)*, pages 764–777. Association for Computational Linguistics.

A Annotation of Animal Farm

The text of Animal Farm is accessed from the site of Project Gutenberg Australia.[5] We describe the annotation process as follows.

Preprocessing We append the chapter titles in front of the texts of the chapters as the first sentences of the chapters, and concatenate the texts of the chapters into a single document. Before the annotation, we adopt a character list containing the 20 characters as the 20 entities,[6] and first extract the their mentions with string matching.

Annotation Tool We developed a web-based annotation tool with a server developed with Flask[7] as shown in Figure 8. The entities are listed in the top of the web page, shown in the blue boxes with the texts from the character list. The text of the book are splitted into multiple segments so we can view each segment in one page, and the annotator can switch the segment by the "Previous" and "Next" buttons. In each page, the segment of text is splitted into multiple paragraphs to avoid getting lost in the long text. When we click one entity in the blue box, e.g. "Mr. Jones", it means we are now annotating this entity now, and the mentions of this entity in the text gets gray colored. We annotate the mentions by clicking the words in the text. Once the word is clicked, it gets gray colored (e.g. "his", "he", "himeself"). Consecutive words are treated as one span of mention. When finished annotation for the current page, we click the "Save" button to save the annotation to server.

Annotation Standard The mentions of the entities include pronouns (e.g. "he", "she", "I" "they", etc), possessive pronouns (e.g. "his", "her", "my", "their", etc) as the case in Ontonotes 5.0, and other noun phrases (e.g. "Jones", "Major", "the three dogs", etc).

The annotation is done by a human expert for 5 hours and there are totally 1,722 mentions annotated.

B Data and Model license

Following the instruction of ACL, we list the science artifacts used in this work in Table 7.

[5]The book is available at https://gutenberg.net.au/ebooks01/0100011h.html

[6]We adapt the character list from https://www.litcharts.com/lit/animal-farm/characters

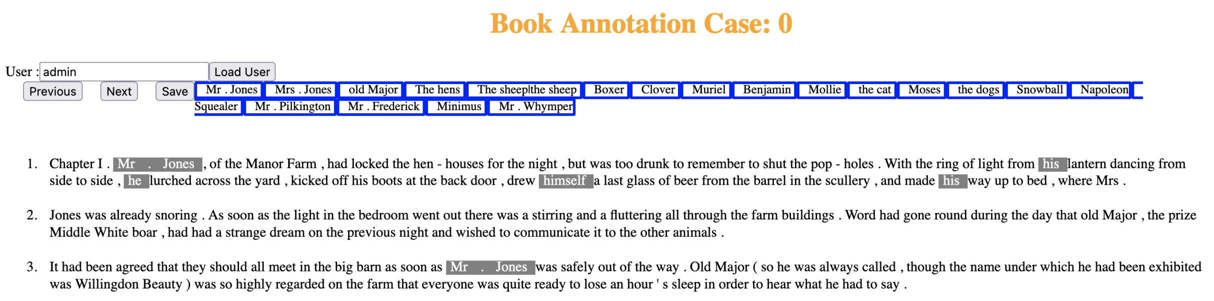
[7]https://flask.palletsprojects.com/en/2.1.x/

Dataset	License
OntoNotes	LDC
LongtoNotes	CC 4.0 + LDC
LitBank	CC 4.0
WikiCoref	No license stated
Toshniwal et al. (2021)	No license stated
Longformer	Apache-2.0

Table 7: License of science artifacts.

C More Metrics

Besides the F1 score that we reported in the main text, we give results that are evaluated by commonly used metrics for the comparison with others, including MUC (Vilain et al., 1995), B-CUBED (Bagga and Baldwin, 1998), and CEAFE (Luo, 2005). For better visualization, we split results into three tables, Table 8, Table 9, and Table 10. There metrics show the same trend of F1 score that are shown in the main text.



1. Chapter I . Mr . Jones , of the Manor Farm , had locked the hen - houses for the night , but was too drunk to remember to shut the pop - holes . With the ring of light from his lantern dancing from side to side , he lurched across the yard , kicked off his boots at the back door , drew himself a last glass of beer from the barrel in the scullery , and made his way up to bed , where Mrs .
2. Jones was already snoring . As soon as the light in the bedroom went out there was a stirring and a fluttering all through the farm buildings . Word had gone round during the day that old Major , the prize Middle White boar , had had a strange dream on the previous night and wished to communicate it to the other animals .
3. It had been agreed that they should all meet in the big barn as soon as Mr . Jones was safely out of the way . Old Major (so he was always called , though the name under which he had been exhibited was Willingdon Beauty) was so highly regarded on the farm that everyone was quite ready to lose an hour ' s sleep in order to hear what he had to say .

Figure 8: The annotation tool for annotating the book Animal Farm.

		LitBank		
Method	Cache Size	MUC	B^3	CEAFE
Dual cache	25+25	87.7 $\pm$ 0.28	78.8 $\pm$ 0.72	69.8 $\pm$ 0.04
Dual cache	100+100	88.1 $\pm$ 0.28	79.1 $\pm$ 0.28	70.8 $\pm$ 0.03
Dual cache	250+250	88.2 $\pm$ 0.31	79.2 $\pm$ 0.71	71.0 $\pm$ 0.13

Table 8: Results on LitBank using three different metrics.

		OntoNotes		
Method	Cache Size	MUC	B^3	CEAFE
Dual cache	25+25	85.3 $\pm$ 0.10	78.9 $\pm$ 0.23	74.5 $\pm$ 0.29
Dual cache	100+100	86.2 $\pm$ 0.09	80.2 $\pm$ 0.22	76.7 $\pm$ 0.23
Dual cache	250+250	86.3 $\pm$ 0.09	80.3 $\pm$ 0.22	76.8 $\pm$ 0.25

Table 9: Results on OntoNotes using three different metrics.

		WikiCoref		
Method	Cache Size	MUC	B^3	CEAFE
Dual cache	25+25	70.2	58.8	50.7
Dual cache	100+100	71.7	61.6	54.1
Dual cache	250+250	72.1	62.1	54.7

Table 10: Results on WikiCoref using three different metrics.

		LongtoNotes			
Method	Cache Size	F1	MUC	B^3	CEAFE
Dual cache	25+25	76.2 $\pm$ 0.14	84.3 $\pm$ 0.23	74.4 $\pm$ 0.22	70.1 $\pm$ 0.09
Dual cache	100+100	77.7 $\pm$ 0.11	85.2 $\pm$ 0.10	75.9 $\pm$ 0.21	72.1 $\pm$ 0.03
Dual cache	250+250	77.9 $\pm$ 0.13	85.2 $\pm$ 0.12	76.3 $\pm$ 0.24	72.2 $\pm$ 0.04

Table 11: Results on LongtoNotes using three different metrics.

		LongtoNotes$_S$			
Method	Cache Size	F1	MUC	B^3	CEAFE
Dual cache	25+25	74.8 $\pm$ 0.11	85.2 $\pm$ 0.25	72.0 $\pm$ 0.21	67.1 $\pm$ 0.07
Dual cache	100+100	76.3 $\pm$ 0.10	85.9 $\pm$ 0.09	73.9 $\pm$ 0.20	69.3 $\pm$ 0.03
Dual cache	250+250	76.3 $\pm$ 0.08	85.7 $\pm$ 0.11	74.2 $\pm$ 0.27	69.2 $\pm$ 0.03

Table 12: Results on the long-doc subset LongtoNotes$_S$ using three different metrics.