Active Learning for Coreference Resolution using Discrete Annotation

Belinda Z. Li[†*] **Gabriel Stanovsky**[♠◇] **Luke Zettlemoyer**[♠†]
[♠] University of Washington [◇]Allen Institute for AI [†]Facebook
belindali@fb.com
{gabis,lsz}@cs.washington.edu

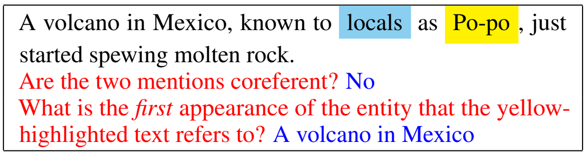
Abstract

We improve upon pairwise annotation for active learning in coreference resolution, by asking annotators to identify mention antecedents if a presented mention pair is deemed not coreferent. This simple modification, when combined with a novel mention clustering algorithm for selecting which examples to label, is much more efficient in terms of the performance obtained per annotation budget. In experiments with existing benchmark coreference datasets, we show that the signal from this additional question leads to significant performance gains per human-annotation hour. Future work can use our annotation protocol to effectively develop coreference models for new domains. Our code is publicly available.[1]

1 Introduction

Coreference resolution is the task of resolving anaphoric expressions to their antecedents (see Figure 1). It is often required in downstream applications such as question answering (Dasigi et al., 2019) or machine translation (Stanovsky et al., 2019). Exhaustively annotating coreference is an expensive process as it requires tracking coreference chains across long passages of text. In news stories, for example, important entities may be referenced many paragraphs after their introduction.

Active learning is a technique which aims to reduce costs by annotating samples which will be most beneficial for the learning process, rather than fully labeling a large fixed training set. Active learning consists of two components: (1) a task-specific learning algorithm, and (2) an iterative sample selection algorithm, which examines the performance of the model trained at the previous iteration and selects samples to add to the annotated training set. This method has proven successful for various tasks in low-resource domains (Garrette and Baldridge, 2013; Kholghi et al., 2015; Syed et al., 2016, 2017).



A volcano in Mexico, known to locals as Po-po, just started spewing molten rock.
Are the two mentions coreferent? No
What is the *first* appearance of the entity that the yellow-highlighted text refers to? A volcano in Mexico

Figure 1: Discrete annotation. The annotator is shown the document, a span (yellow), and the span's predicted antecedent (blue). In case the answer to the coreference question is negative (i.e., the spans are not coreferring), we present a follow-up question ("what is the *first* appearance of the entity?"), providing additional cost-effective signal. Our annotation interface can be seen in Figure 5 in the Appendix.

Sachan et al. (2015) showed that active learning can be employed for the coreference resolution task. They used gold data to simulate pairwise human-annotations, where two entity mentions are annotated as either coreferring or not (see first question in Figure 1).

In this paper, we propose two improvements to active learning for coreference resolution. First, we introduce the notion of *discrete annotation* (Section 3), which augments pairwise annotation by introducing a simple additional question: if the user deems the two mentions non-coreferring, they are asked to mark the first occurrence of one of the mentions (see second question in Figure 1). We show that this simple addition has several positive implications. The feedback is relatively easy for annotators to give, and provides meaningful signal which dramatically reduces the number of annotations needed to fully label a document.

Second, we introduce *mention clustering* (Section 4). When selecting the next mention to label, we take into account aggregate model predictions

*Work done while at the University of Washington.

[1] https://github.com/belindal/discrete-active-learning-coref

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8320–8331
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

for all antecedents which belong to the same cluster. This avoids repeated labeling that would come with separately verifying every mention pair within the same cluster, as done in previous methods.

We conduct experiments across several sample selection algorithms using existing gold data for user labels and show that both of our contributions significantly improve performance on the CoNLL-2012 dataset (Pradhan et al., 2012). Overall, our active learning method presents a superior alternative to pairwise annotation for coreference resolution, achieving better performing models for a given annotation budget.

2 Background

Our work relies on two main components: a coreference resolution model and a sample selection algorithm.

Coreference resolution model We use the span ranking model introduced by Lee et al. (2017), and later implemented in AllenNLP framework (Gardner et al., 2018). This model computes span embeddings for all possible spans i in a document, and uses them to compute a probability distribution $P(y = \text{ant}(i))$ over the set of all candidate antecedents $\mathcal{Y}(i) = \{K$ previous mentions in the document$\} \cup \{\epsilon\}$, where ϵ is a dummy antecedent signifying that span i has no antecedent. This model does not require additional resources, such as syntactic dependencies or named entity recognition, and is thus well-suited for active learning scenarios for low-resource domains.

Sample selection algorithm Previous approaches for the annotation of coreference resolution have used mostly *pairwise selection*, where pairs of mentions are shown to a human annotator who marks whether they are co-referring (Gasperin, 2009; Laws et al., 2012; Zhao and Ng, 2014; Sachan et al., 2015). To incorporate these binary annotations into their clustering coreference model, Sachan et al. (2015) introduced the notion of *must-link* and *cannot-link* penalties, which we describe and extend in Section 4.

3 Discrete Annotation

In *discrete annotation*, as exemplified in Figure 1, we present the annotator with a document where the least certain span i ("Po-po", in the example) and i's model-predicted antecedent, $A(i)$ ("locals"), are highlighted. Similarly to pairwise annotation, annotators are first asked whether i and $A(i)$ are coreferent. If they answer positively, we move on to the next sample. Otherwise, we deviate from pairwise sampling and ask the annotator to mark the antecedent for i ("A volcano in Mexico") as the *follow-up* question.[2] The annotator can abstain from answering the follow-up question in case i is not a valid mention or if it does not have an antecedent in the document. See Figure 5 in the Appendix for more example annotations.

In Section 5, we show that discrete annotation is superior to the classic pairwise annotation in several aspects. First, it makes better use of human annotation time, as often an annotator needs to resolve the antecedent of the presented mention to answer the first question. For example, identifying that "Po-po" refers to the volcano, and not the locals. Second, we find that discrete annotation is a better fit for mention ranking models (Lee et al., 2017), which assign the most-likely antecedent to each mention, just as an annotator does in discrete annotation.

4 Mention Clustering

We experiment with three selection techniques by applying popular active learning selectors like entropy or query-by-committee (Settles, 2010) to *clusters* of spans. Because our model outputs antecedent probabilities and predictions, we would like to aggregate these outputs, such that we have only one probability per mention cluster rather than one per antecedent. We motivate this with an example: suppose span i's top two most likely antecedents are y_1 and y_2. In scenario 1, y_1 and y_2 are predicted to be clustered together, and in scenario 2, they are predicted to be clustered apart. Span i should have a "higher certainty" in scenario 1 (and thus be less likely to be picked by active learning), because its two most likely antecedents both imply the same clustering, whereas in scenario 2, picking y_1 vs. y_2 results in a different downstream clustering. Thus, rather than simply using the raw probability i refers to a particular antecedents, we use the probability i *belongs to a certain cluster*. This implies modelling y_1 and y_2 "jointly" in scenario 1, and separately in scenario 2.

Formally, we compute the probability that a span i belongs in a cluster C by summing $P(\text{ant}(i) = y)$

[2] For consistency, we ask annotators to select the *first* antecedent of i in the document.

for all y that belong in some cluster C, since i having an antecedent in a cluster necessarily also implies i is also in that cluster. This allows us to convert the predicted antecedent probabilities to in-cluster probabilities:

$$P(i \in C) = \sum_{y \in C \cap \mathcal{Y}(i)} P(\text{ant}(i) = y) \tag{1}$$

Similarly, for query-by-committee, we aggregate predictions such that we have one vote per cluster rather than one vote per antecedent:

$$V(i \in C) = \sum_{y \in C \cap \mathcal{Y}(i)} V(A(i) = y) \tag{2}$$

where $V(A(i) = y) \in \{0, 1, \cdots, \mathcal{M}\}$ refers to the number of models that voted y to be the antecedent of i.

The cluster information ($y \in C \cap \mathcal{Y}(i)$) we use in Equations 1 and 2 is computed from a combination of model-predicted labels and labels queried through active learning. Antecedents which were not predicted to be in clusters are treated as singleton clusters.

Additionally, to respect user annotations during the selection process, we must keep track of all prior annotations. To do this, we use the concept of *must-link* (ML; if two mentions are judged coreferent) and *cannot-link* (CL; if two mentions are judged non-coreferent) relations between mentions introduced by Sachan et al. (2015), and adapt it for our purposes. Specifically, in our discrete setting, we build the links as follows: if the user deems the pair coreferent, it is added to ML. Otherwise, it is added to CL, while the user-corrected pair (from the second question) is always added to ML.

In addition, we use these links to guide how we *select* for the next mention to query. For example, if a CL relation exists between spans m_1 and m_2, we will be less likely to query for m_1, since we are slightly more certain on what m_1's antecedent should be (not m_2). Formally, we revise probabilities and votes $P(i \in C)$ and $V(i \in C)$ in accordance to our link relations, which affects the selector uncertainty scores.[3]

Finally, following (Sachan et al., 2015), we impose transitivity constraints, which allow us to model links beyond what has been explicitly pointed out during annotation:

$$ML(m_i, m_j) \wedge ML(m_j, m_k) \rightarrow ML(m_i, m_k) \tag{3}$$

$$CL(m_i, m_j) \wedge ML(m_i, m_k) \rightarrow CL(m_j, m_k) \tag{4}$$

However, recomputing these closures after each active learning iteration can be extremely inefficient. Instead, we build up the closure incrementally by adding only the minimum number of necessary links to maintain the closure every time a new link is added.

We experiment with the following clustered selection techniques:

Clustered entropy We compute entropy over cluster probabilities and select the mention with the highest *clustered* entropy:

$$E(i) = - \sum_{C \in \text{all clusters}} P(i \in C) \cdot \log P(i \in C) \tag{5}$$

Where $P(i \in C)$ is defined as in Equation 1.

Clustered query-by-committee We train $\mathcal{M}$ models (with different random seeds) and select the mention with the highest *cluster* vote entropy:

$$\text{VE}(i) = - \sum_{C \in \text{all clusters})} \frac{V(i \in C)}{\mathcal{M}} \cdot \log \frac{V(i \in C)}{\mathcal{M}} \tag{6}$$

Using votes counted over clusters, as defined in Equation 2.

Least coreferent clustered mentions / Most coreferent unclustered mentions (LCC/MCU) We aim to select a subset of spans for which the model was least confident in its prediction. For each span i which was assigned a cluster C_i, we compute a score $s_C(i) = P(i \in C_i)$, and choose n spans with the smallest $s_C(i)$. For each singleton j, we give an "unclustered" score $s_U(i) = \max_{C \in \text{all clusters}} P(j \in C)$ and choose m spans with the *largest* $s_U(i)$. $P(i \in C_i)$ and $P(j \in C)$ are computed with Equation 1.

5 Evaluation

We compare discrete versus pairwise annotation using the English CoNLL-2012 coreference dataset (Pradhan et al., 2012). Following Sachan et al. (2015), we conduct experiments where user judgments are simulated from gold labels.

[3] See Section A.2 in the appendix for more details.

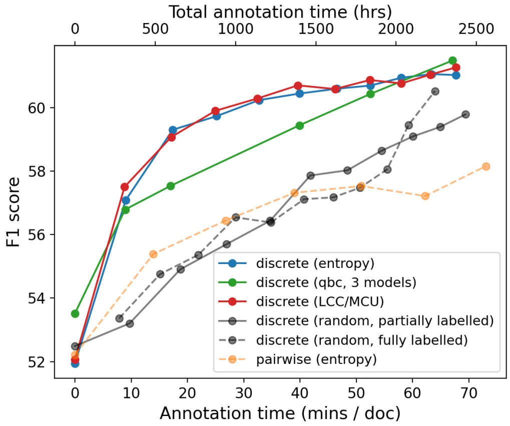


Figure 2: Comparing various selectors for discrete versus pairwise annotation (dashed orange line).

Set	# labels/doc	Active learning iteration	# docs	# ?s
A	20	1st (retrained 0x)	5	15
	20	7th (retrained 6x)	5	15
	200	2nd (retrained 1x)	5	15
	200	8th (retrained 7x)	5	15
B	20	2nd (retrained 1x)	5	15
	20	8th (retrained 7x)	5	15
	200	1st (retrained 0x)	5	15
	200	7th (retrained 6x)	5	15

Table 1: Timing experiments sampling. For each of the 2 datasets, we collected 60 total active learning questions from 20 documents. We collected 5 documents and 15 questions for each of the 4 categories: trained with many/few labels per document, and early/late in active learning process. The 15 questions were sampled randomly from within an iteration.

Annotation time estimation To compare annotation times between pairwise and discrete questions, we collected eight 30-minute sessions from 7 in-house annotators with background in NLP. Annotators were asked to answer as many instances as they could during those 30 minutes. We additionally asked 1 annotator to annotate *only* discrete questions for 30 minutes. To be as representative as possible, the active learning queries for these experiments were sampled from various stages of active learning (see Table 1). On average, an annotator completed about 67 questions in a single session, half of which were answered negatively, requiring the additional discrete question. Overall, these estimates rely on 826 annotated answers. Our annotation interface is publicly available,[4] see examples in Figure 5 in the Appendix.

Timing results are shown in Table 2. Answering the discrete question after the initial pairwise question takes about the same time as answering the first question (about $16s$). Furthermore, answering only discrete questions took $28.01s$ per question, which confirmed that having an initial pairwise question indeed saves annotator time if answered positively.

	Avg. Time per ?
Initial question	$15.96s$
Follow-up question	$15.57s$
ONLY Follow-up questions	$28.01s$

Table 2: Average annotation time for the initial pairwise question, the discrete followup question, and the discrete question on its own.

In the following experiments, we use these measurements to calibrate pairwise and discrete followup questions when computing total annotation times.

Baselines We implement a baseline for pairwise annotation with entropy selector. We also implement two discrete annotation baselines with random selection. The *partially-labelled* baseline follows the standard active learning training loop, but selects the next mention to label at random. The *fully-labelled* baseline creates a subset of the training data by taking as input an annotation time t and selecting at random a set of documents that the user can *fully* label in t hours using ONLY discrete annotation. By comparing the fully-labelled baseline against our active learning results, we can determine whether active learning is effective over labelling documents exhaustively .

Hyperparameters We use the model hyperparameters from the AllenNLP implementation of Lee et al. (2017). We train up to 20 epochs with a patience of 2 before adding labels. After all documents have been added, we retrain from scratch. We use a query-by-committee of $\mathcal{M} = 3$ models, due to memory constraints. For LCC/MCU, given L annotations per document, we split the annotations equally between clusters and singletons.

Results Figure 2 plots the performance of discrete annotation with the various selectors from Section 4, against the performance of pairwise annotation, calibrated according to our timing experiments. In all figures, we report MUC, B3, and CEAFe as an averaged F1 score.

The three non-random active learning frameworks outperform the fully-labelled baseline, show-

[4] https://belindal.github.io/timing_experiments

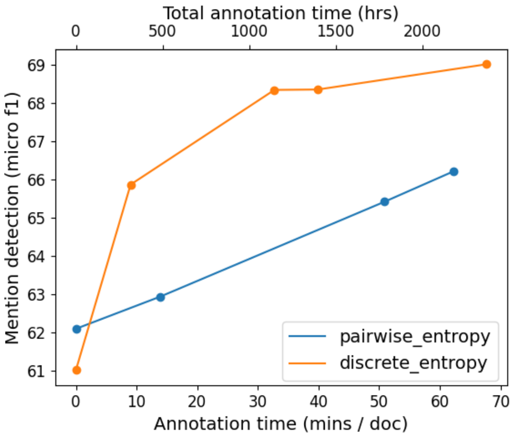


Figure 3: Mention detection accuracy (in document-micro F1) for pairwise versus discrete selection per human annotation time.

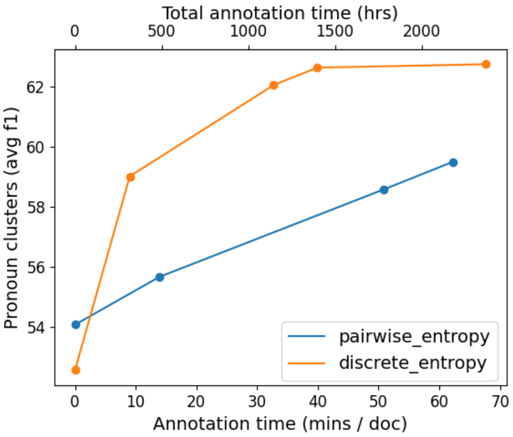


Figure 4: Pronoun resolution accuracy (average F1) for pairwise versus discrete selection per human annotation time.

ing that active learning is more effective for coreference resolution when annotation budget is limited.

Most notably, Figure 2 shows that every non-random discrete selection protocol outperforms pairwise annotation. Where the gap in performance is the largest ($>$ 15 minutes per document), we consistently improve by $\sim$4% absolute $F1$ over pairwise selection.

6 Analysis

A major reason discrete annotation outperforms the pairwise baseline is that the number of pairwise annotations needed to fully label a document is much larger than the number of discrete annotations. In an average development document with 201 candidates per mention, the number of pairwise queries needed to fully label a document is $15,050$, while the maximum number of discrete queries is only 201 (i.e., asking for the antecedent of every mention). Thus, the average document can be fully annotated via discrete annotation in only 2.6% of the time it takes to fully label it with pairwise annotation, suggesting that our framework is also a viable *exhaustive* annotation scheme.

Further analysis shows that the improvement in discrete selection stems in part from better use of annotation time for mention detection accuracy (Figure 3) and pronoun resolution (Figure 4), in which we measure performance only on clusters with pronouns, as identified automatically by the spaCy tagger (Honnibal and Montani, 2017) .

Finally, Table 3 shows ablations on our discrete annotation framework, showing the contribution of each component of our paradigm.

	F1 score
Discrete annotation	57.08
−clustered probabilities	56.49
−incremental link closures	56.98
Pairwise annotation	54.27

Table 3: Ablations over the different model elements, at a single point ($\sim$315 annotation hours). Entropy selector was used for all experiments.

7 Discussion and Conclusion

We presented discrete annotation, an attractive alternative to pairwise annotation in active learning of coreference resolution in low-resource domains. By adding a simple question to the annotation interface, we obtained significantly better models per human-annotation hour. In addition, we introduced a clustering technique which further optimizes sample selection during the annotation process. More broadly, our work suggests that improvements in annotation interfaces can elicit responses which are more efficient in terms of the obtained performance versus the invested annotation time.

Acknowledgements

We would like to thank Christopher Clark, Terra Blevins, and the anonymous reviewers for their helpful feedback, and Aaron Jaech, Mason Kamb, Madian Khabsa, Kaushal Mangipudi, Nayeon Lee, and Anisha Uppugonduri for their participation in our timing experiments.

References

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Caroline Gasperin. 2009. Active learning for anaphora resolution. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Mahnoosh Kholghi, Laurianne Sitbon, Guido Zuccon, and Anthony Nguyen. 2015. Active learning: a step towards automating medical concept extraction. *Journal of the American Medical Informatics Association*, 23(2):289–296.

Florian Laws, Florian Heimerl, and Hinrich Schütze. 2012. Active learning for coreference resolution. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 508–512, Montréal, Canada. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. *ArXiv*, abs/1707.07045.

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

Mrinmaya Sachan, Eduard Hovy, and Eric P. Xing. 2015. An active learning approach to coreference resolution. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1312–1318. AAAI Press.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *ACL*, page (to appear), Florence, Italy. Association for Computational Linguistics.

A. R. Syed, A. Rosenberg, and E. Kislal. 2016. Supervised and unsupervised active learning for automatic speech recognition of low-resource languages. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5320–5324.

A. R. Syed, A. Rosenberg, and M. Mandel. 2017. Active learning for low-resource speech recognition: Impact of selection size and language modeling data. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5315–5319.

Shanheng Zhao and Hwee Tou Ng. 2014. Domain adaptation with active learning for coreference resolution. In *Proceedings of the 5th International Workshop on Health Text Mining and Information Analysis (Louhi)*, pages 21–29, Gothenburg, Sweden. Association for Computational Linguistics.

A Appendix

A.1 Timing Experiment Details and Computations.

In order to properly calibrate the results from discrete and pairwise querying, we conducted experiments (eight 30-minute sessions) to time how long annotators take to answer discrete and pairwise questions. See Figure 5 for the interface we designed for our experiments.

The questions we ask for the experiment are all sampled from real queries from full runs of our active learning simulations. To obtain representative times, we sampled a diverse selection of active learning questions–at various stages of active learning (first iteration before retraining vs. after retraining n times) and various numbers of annotation per document (20 vs. 200). For each document, we randomly selected between 1-5 questions (of the total 20 or 200) to ask the annotator. Full details on how we sampled our queries can be found in Table 1. Note that we divided our samples into two datasets. We ran four 30-minute sessions with Dataset A before Dataset B and four 30-minute sessions with Dataset B before Dataset A–for a total of eight 30-minute sessions across 7 annotators (1 annotator completed a 1-hour session).

Since pairwise annotation is the same as answering only the initial question under the discrete setting, we run a single discrete experiment for each annotation session and use the time taken to answer an initial question as a proxy for pairwise annotation time. Our results show that answering the initial question took an average of $15.96s$ whereas answering the follow-up question took $15.57s$. Thus, we derive the following formulas to compute the time it takes for pairwise and discrete annotation:

$$t = 15.96p \tag{7}$$

$$t = 15.96d_c + 15.57d_{nc} \tag{8}$$

where $p = \#$ of pairwise instances. $d_c, d_{nc} = \#$ of discrete instances for which the initial pair was "coreferent" (d_c) and "not coreferent" (d_{nc}), respectively. We also compute the number of pairwise examples p we can query in the same time it takes to query $d_c + d_{nc}$ discrete examples:

$$\begin{aligned} 15.96p &= 15.96d_c + 15.57d_{nc} \\ p &= d_c + 0.976d_{nc} \end{aligned} \tag{9}$$

Moreover, we additionally conduct a single 30-minute experiment to determine how long it takes to answer *only* discrete questions (without the initial pairwise step). We find that it takes $28.01s$ per question under the only-discrete setting. This is longer than the time it takes to answer a pairwise question, thus confirming that having an initial pairwise question indeed saves time if the pair is coreferent. Moreover, this also shows that answering the initial pairwise question significantly helps with answering the follow-up discrete question.

A.2 Additional Model Adaptations

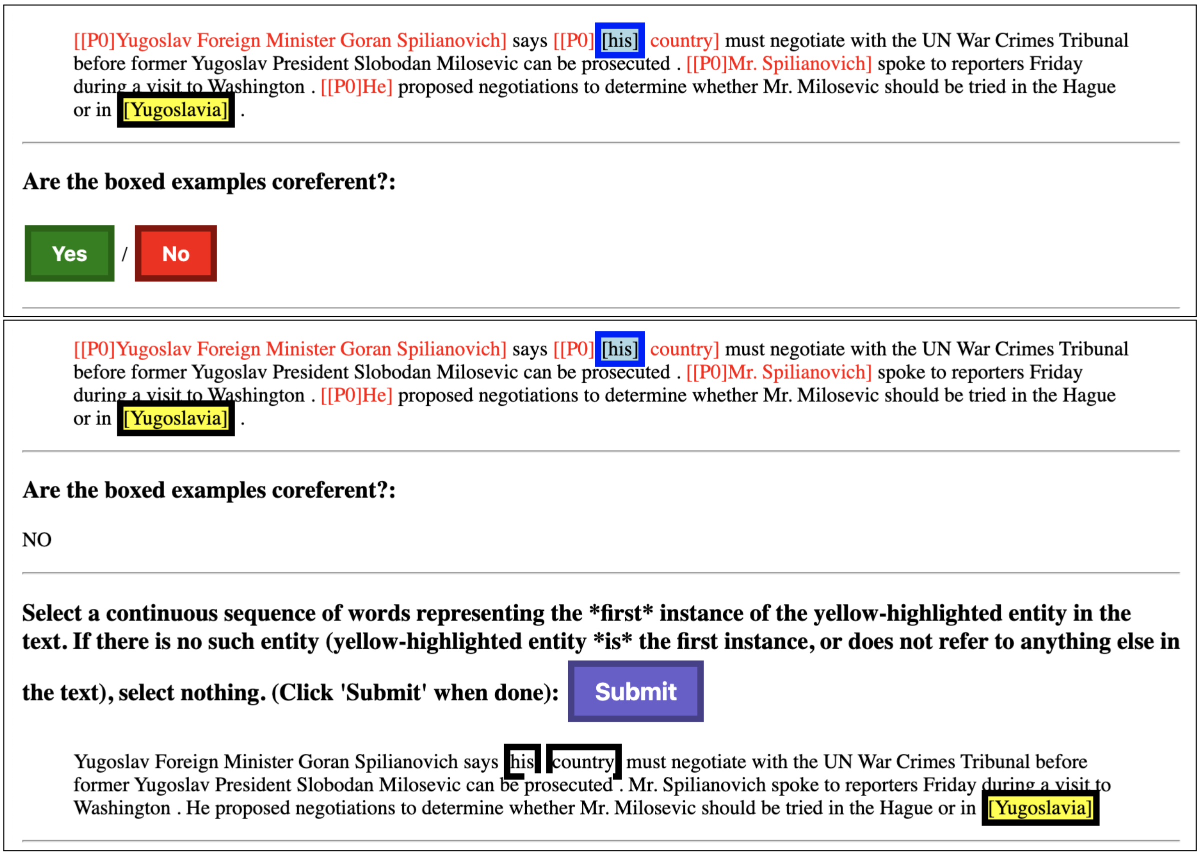
Adapting Link Relations for our Model We use must-link and cannot-link relations between mentions to guide our active learning selector. We revise probabilities and model outputs (from which the model computes uncertainty scores for entropy, QBC, and LCC/MCU) in accordance to the following rules:

1. **Clustered entropy**. For every $CL(a, b)$ relationship, we set $P(\text{ant}(a) = b) = 0$ and re-normalize probabilities of all other candidate antecedents. This decreases the probability that the active learning selector chooses a. Moreover, for every $ML(a, b)$ relationship, we set $P(\text{ant}(a) = b) = 1$ and $P(\text{ant}(a) = c) = 0$ for all $c \neq b$. If there are multiple ML relationships involving a, we choose only one of a's antecedent to set to 1 (to maintain the integrity of the probability distribution). This guarantees that the active learning selector will never select a, as any ML link out of a means we have already queried for a.

2. **Clustered query-by-committee**. To ensure we do not choose a mention we have already queried for, after each user judgment, for every $ML(a, b)$ relation, we set $V(A(a) = b) = \mathcal{M}$, and $V(A(a) = c) = 0$ for all other $c \neq b$. Moreover, for every $CL(a, b)$ relation, we set $V(A(a) = b) = 0$, which decreases the vote entropy of a, making it less likely for the selector to choose a.

3. **LCC/MCU**. We revise the probabilities in the same way as in clustered entropy and add the constraint that, when choosing MCU spans j, we disregard those that already have probability 1 (signifying that we have already queried for them).

Incremental Closures Algorithm We introduce an algorithm to compute link closures *incrementally*. Instead of re-computing and re-adding the



[[P0]Yugoslav Foreign Minister Goran Spilianovich] says [[P0] [his] country] must negotiate with the UN War Crimes Tribunal before former Yugoslav President Slobodan Milosevic can be prosecuted . [[P0]Mr. Spilianovich] spoke to reporters Friday during a visit to Washington . [[P0]He] proposed negotiations to determine whether Mr. Milosevic should be tried in the Hague or in [Yugoslavia] .

Are the boxed examples coreferent?:

[[P0]Yugoslav Foreign Minister Goran Spilianovich] says [[P0] [his] country] must negotiate with the UN War Crimes Tribunal before former Yugoslav President Slobodan Milosevic can be prosecuted . [[P0]Mr. Spilianovich] spoke to reporters Friday during a visit to Washington . [[P0]He] proposed negotiations to determine whether Mr. Milosevic should be tried in the Hague or in [Yugoslavia] .

Are the boxed examples coreferent?:

NO

Select a continuous sequence of words representing the *first* instance of the yellow-highlighted entity in the text. If there is no such entity (yellow-highlighted entity *is* the first instance, or does not refer to anything else in the text), select nothing. (Click 'Submit' when done):

Yugoslav Foreign Minister Goran Spilianovich says his country must negotiate with the UN War Crimes Tribunal before former Yugoslav President Slobodan Milosevic can be prosecuted . Mr. Spilianovich spoke to reporters Friday during a visit to Washington . He proposed negotiations to determine whether Mr. Milosevic should be tried in the Hague or in [Yugoslavia]

Figure 5: Timing experiments interface. Top: The initial pairwise question. Bottom: The user is presented with the discrete question when they click “No”. They are asked to select the appropriate tokens in the text representing the first occurrence of the yellow entity in the text.

entire set of closures (based on a set of all prior human annotations that we keep track of) each time we query for a new mention, we add the minimum set of necessary links. See Algorithm 1.

To determine how much time our incremental closure algorithm saves over recomputing closures from scratch, we simulated annotations on a single document with 1600 mentions, and recorded how long it took to re-compute the closure after each annotation. Our experiments show that recomputing from scratch takes progressively longer as more labels get added: at 1600 labels, our incremental algorithm is 556 times faster than recomputing from scratch ($1630ms$ vs. $2.93ms$).

Figure 6 plots the runtime of our incremental closure algorithm (“incremental closure”) against the run-time of recomputing closures from scratch (“closure”) using Equations 3 and 4. In the latter case, we keep track of the set of user-added edges which we update after each annotation, and re-compute the closures from that set.

A.3 Additional Analysis

Computing the time to fully-label a document under discrete and pairwise annotation. First, we compute the maximum number of pairwise questions we can ask. We consider the setup of Lee et al. (2017)’s model. This model considers only spans with highest mention scores (the “top spans”), and only considers at most K antecedents per top span. Thus, for a document with m top spans, we can ask up to

$$\frac{K(K-1)}{2} + (m-K)K \tag{10}$$

pairwise questions. The first factor $\frac{K(K-1)}{2}$ comes from considering the first K spans in the document. For each of these spans $i = 1 \cdots K$, we can ask about the first $i-1$ spans. The second factor $(m-K)K$ comes from considering the spans after the K-th span. For each of these $m-K$ spans in the document, we can only consider up to K antecedents. Using statistics for the average document ($m = 201$) and the standard hyper-parameter settings ($K = 100$), we plug into Equation 10 to

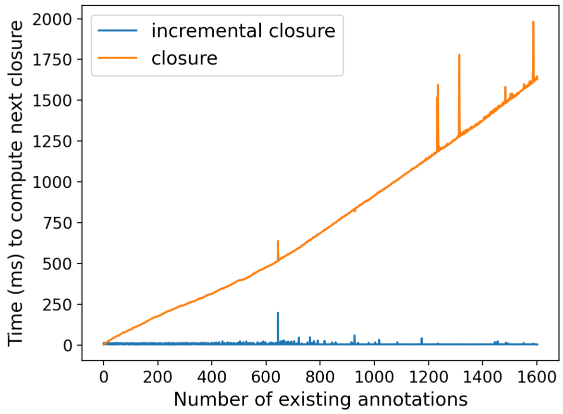


Figure 6: Under each closure algorithm, the time to compute the closure after the next annotation is added, as # of existing annotations increases.

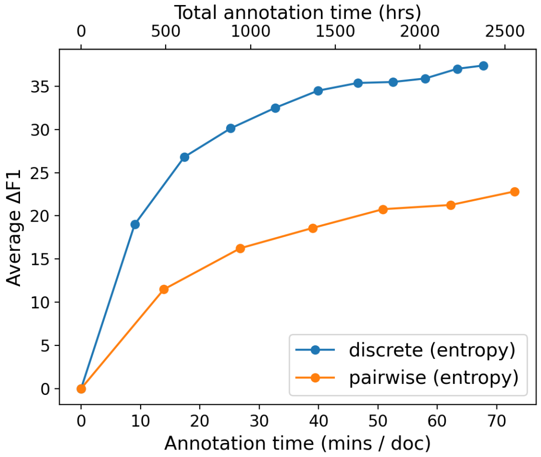


Figure 7: Comparing F1 score improvement on $\overline{D_U}$ for discrete vs. pairwise annotation.

get $15,050$ overall pairwise questions needed to fully label a document (in worst-case). Meanwhile, the maximum number of discrete questions we can ask is only 201 (i.e., asking for the antecedent of every mention). Using timing Equations 7 and 8, we compute that it takes at most $6337.53s$ to answer 201 discrete questions in the worst-case scenario, and $240198s$ to answer 15050 pairwise questions. Thus, in the worst-case scenario for both discrete and pairwise selection, discrete selection will take only 2.64% of the time it takes pairwise selection to fully label a document.

Quantifying "Information Gain" from Discrete and Pairwise Annotation. Let $\overline{D_U}$ be the set of training documents we are *annotating for* in a given round of active learning. To better quantify how much information discrete and pairwise annotation can supply in same amount of time, we define $\Delta F1$ as the change in the $F1$ score on $\overline{D_U}$, before and after model predictions are supplemented with user annotation.

Figure 7 shows average $\Delta F1$ as annotation time increases for discrete and pairwise annotation. Across the 10 annotation times we recorded, discrete annotation results in an average $\Delta F1$ that more than twice that of pairwise, in the same annotation time.

A.4 Hyperparameters

Model. We preserve the hyperparameters from the AllenNLP implementation of Lee et al. (2017)'s model. The AllenNLP implementation mostly maintains the original hyperparameters, except it sets the maximum number of antecedents considered to $K = 100$, and excludes speaker features and variational dropout, due to machine memory limitations.

Training. We use a 700/2102 fully-labelled/unlabelled initial split of the training data, and actively label 280 documents at a time. We train to convergence each round. Before all documents have been added, we train up to 20 epochs with a patience of 2 before we add more training documents. After all documents have been added, we retrain from scratch and use the original training hyperparameters from Lee et al. (2017).

Selectors. For query-by-committee, we use a committee of $\mathcal{M} = 3$ models. We were not able to experiment with more due to memory constraints.

For LCC/MCU, given L annotations per document, we allocate n annotations to least-coreferent clustered mentions and the remaining m to most-coreferent unclustered mentions. We use $n = \min(L/2, \text{number of clustered spans})$, and $m = \min(L - n, \text{number of un-clustered spans})$.

A.5 Active Learning Training Setup Full Details

In our active learning setup, we begin by training our model on a 700-document subset of the full training set. We discard the labels of the remaining 2102 documents. In each round of active learning, we choose 280 unlabelled documents, and query up to Q annotations per document. We then add these documents to the labelled set and continue training our model on this set (now with new documents). After all documents have been labelled, we retrain our model on the full document set from scratch, resetting all model and trainer parameters.

In Algorithm 2, we show our main training loop for active learning using discrete selection. This is the training loop we use for our clustered entropy and LCC/MCU selectors, and our partially-labelled random baseline. In Algorithm 3, we modify that loop for the clustered query-by-committee selector.

In Algorithm 1, we show our incremental closures algorithm, which builds up the transitive closure incrementally by adding only the minimum number of necessary links to maintain the closure each time a new link is added.

Algorithm 1: Incremental Link Closures Algorithm

Let (a, b) = link pair being added, $A = a$'s old cluster before the pair is added, $B = b$'s old cluster before the pair is added, $\overline{A}$ = set of element a has a CL relationship to before the pair is added, $\overline{B}$ = set of elements b has a CL relationship to before the pair is added.

1. If pair (a, b) was added to *must-link*, both must-link and cannot-link needs to be updated.
   First, resolve the MLs by adding a ML relationship between every element in A and every element in B:

   $$\forall a', b' \quad (ML(a, a') \wedge ML(b, b')) \rightarrow (ML(a, b') \wedge ML(a', b) \wedge ML(a', b'))$$

   Next, resolve the CLs by adding a CL relationship between every element of A and $\overline{B}$, and every element of B and $\overline{A}$:

   $$\forall a', \hat{b} \quad (ML(a, a') \wedge CL(b, \hat{b})) \rightarrow (CL(a, \hat{b}) \wedge CL(a', \hat{b}))$$

   $$\forall b', \hat{a} \quad (ML(b, b') \wedge CL(a, \hat{a})) \rightarrow (CL(b, \hat{a}) \wedge CL(b', \hat{a}))$$

2. If pair (a, b) was added to *cannot-link*, only cannot-link needs to be updated. Add a CL relationship between every element of A and every element of B:

   $$\forall a', b' \quad (ML(a, a') \wedge ML(b, b')) \rightarrow (CL(a, b') \wedge CL(a', b) \wedge CL(a', b'))$$

Algorithm 2: Training loop for active learning

```
$D_F$ = {fully-labelled docs}, $D_U$ = {unlabelled docs}, $D_A$ = {docs labelled through active learning}, $M$ = model, $ML$ =
  must-link pairs, $CL$ = cannot-link pairs;
Init: $D_F$ = {first 700 docs}, $D_U$ = {remaining docs}, $D_A = \emptyset$, $ML = CL = \emptyset$;
while $D_U$ is not empty do
    train $M$ to convergence on data $D_F \cup D_A$;
    $\overline{D_U}$ = 280-document subset of $D_U$;
    for $D \in \overline{D_U}$ do
        $\mathcal{P}_D, \mathcal{L}_D, \mathcal{C}_D$ = run $M$ on $D$;
          $\mathcal{P}_D$ = model-outputted probabilities = $\{P(y = \text{ant}(i)) | y \in \mathcal{Y}(i), i \in \text{top\_spans}(D)\}$
          $\mathcal{L}_D$ = model-outputted antecedent labels = $\{(i, A(i)) | i \in \text{top\_spans}(D)\}$
          $\mathcal{C}_D$ = model-outputted clusters from $\mathcal{L}_D$
        while num_queried < num_to_query do
            $m$ = choose-next-mention-to-query($\mathcal{P}_D, \mathcal{C}_D$);          [[Section 4]]
            $a = \max_{y \in \mathcal{Y}(m) \setminus \epsilon} P(y = \text{ant}(m))$;
            if user deems $m$ and $a$ coreferent then
                $ML = ML \cup (a, m)$;
                $\mathcal{L}_D = \mathcal{L}_D \cup (a, m)$;
                Add $(a, m)$ to $\mathcal{C}_D$;
            else
                $\hat{a}$ = user-selected antecedent for $m$;
                $CL = CL \cup (a, m)$; $ML = ML \cup (\hat{a}, m)$;
                $\mathcal{L}_D = (\mathcal{L}_D \setminus (a, m)) \cup (\hat{a}, m)$;
                Remove $(a, m)$ and add $(\hat{a}, m)$ to $\mathcal{C}_D$;
            end
            $ML, CL$ = compute-link-closures;                       [[Algorithm 1]]
            $\mathcal{P}_D$ = update-based-on-links($ML$, $CL$);            [[Section A.2]]
        end
        Label $D$ with $\mathcal{C}_D$;
    end
    $D_A = D_A \cup \overline{D_U}$; $D_U = D_U \setminus \overline{D_U}$;
end
```

Algorithm 3: Training loop for active learning with QBC selector (Differences from Algorithm 2 are highlighted)

```
$D_F$ = {fully-labelled docs}, $D_U$ = {unlabelled docs}, $D_A$ = {docs labelled through active learning}, $\widehat{M}$ = ensemble
  model of submodels $\{M_1, \cdots, M_{\mathcal{M}}\}$, $ML$ = must-link pairs, $CL$ = cannot-link pairs;
Init: $D_F$ = {first 700 docs}, $D_U$ = {remaining docs}, $D_A = \emptyset$, $ML = CL = \emptyset$;
while $D_U$ is not empty do
    train all $M_1, \cdots, M_{\mathcal{M}}$ to convergence on data $D_F \cup D_A$;
    $\overline{D_U}$ = 280-document subset of $D_U$;
    for $D \in \overline{D_U}$ do
        $\{\mathcal{P}_{D,i}\}, \{\mathcal{L}_{D,i}\}, \mathcal{P}_D, \mathcal{L}_D, \mathcal{C}_D$ = run $\widehat{M}$ on $D$;
          $\mathcal{P}_{D,i}$ = submodel $i$'s output probabilities
          $\mathcal{L}_{D,i}$ = submodel $i$'s output antecedent labels
          $\mathcal{P}_D$ = ensembled (averaged) output probabilities from each submodel
          $\mathcal{L}_D$ = ensembled antecedent labels computed from $\mathcal{P}_D$
          $\mathcal{C}_D$ = ensembled clusters computed from $\mathcal{L}_D$
        while num_queried $<$ num_to_query do
            $m$ = choose-next-mention-to-query($\{\mathcal{L}_{D,i}\}, \mathcal{C}_D$);     [[Section 4]]
            $a = \max_{y \in \mathcal{Y}(m) \setminus \epsilon} P(y = \text{ant}(m))$;     [[Probabilities from $\mathcal{P}_D$]]
            if user deems $m$ and $a$ coreferent then
                $ML = ML \cup (a, m)$;
                Add $(a, m)$ to $\mathcal{C}_D$;
            else
                $\hat{a}$ = user-selected antecedent for $m$;
                $CL = CL \cup (a, m)$; $ML = ML \cup (\hat{a}, m)$;
                Remove $(a, m)$ and add $(\hat{a}, m)$ to $\mathcal{C}_D$;
            end
            $ML, CL$ = compute-link-closures($ML, CL$);     [[Algorithm 1]]
            $\mathcal{L}_{D,i}$ = update-based-on-links($ML$, $CL$);     [[Section A.2]]
        end
        Label $D$ with $\mathcal{C}_D$;
    end
    $D_A = D_A \cup \overline{D_U}$; $D_U = D_U \backslash \overline{D_U}$;
end
```