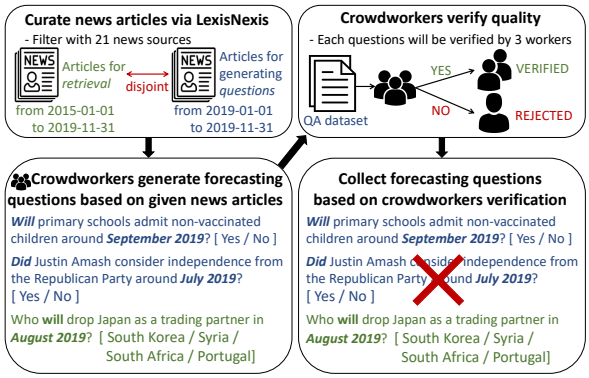

Figure 3: **FORECASTQA generation process.** The input of FORECASTQA creation is a news article corpus and the output is yes-no/multiple-choice questions.

mulation makes our task both a constrained open-domain QA and a forecasting problem—distinct from existing QA tasks.

Challenges in FORECASTQA. Due to the constrained open-domain setting and forecasting properties, testing a model's *forecasting ability* encompasses the following challenges: information retrieval (IR) on limited sources, understanding of temporal and causal relations between events, and finally a forecasting judgement. Our time constraint limits the accessible articles and also creates more challenges than in standard open-domain QA; effective IR methods are necessary to anticipate what knowledge will be useful for predictions from past information sources. Once useful articles have been retrieved, models should understand these articles and reason over pertinent facts from them. Finally, these models use the gleaned knowledge to infer the outcome of a future event. Unlike in other reading comprehension tasks, models cannot rely on the existence of an answer within the text, but must make an educated guess as to what will happen in the future. While our task does encompass reasoning abilities tested in other datasets, no other tasks investigate these reasoning abilities in the context of predicting future events. More analysis on reasoning types can be found in Sec. 4.2.

4 Dataset Construction and Analysis

In this section, we describe how we construct our FORECASTQA dataset and analyze it.

4.1 Construction Details

The data collection is broken down into three sections: (1) gathering a news corpus, (2) generating question-answer-timestamp triples with distractor choices, and (3) verifying the triples' quality. The data generation process is summarized in Fig. 3.

News Corpus Collection. We started by gathering English news articles from LexisNexis[4]. We then curated a list of 21 trustful news sources and filtered articles based on their publishers; we also filtered out non-English articles. Finally, we selected the five-year period of 2015-2019 and filtered out articles outside this period, leaving us with 509,776 articles. This corpus is also used for retrieval in our task setting (*i.e.*, constrained open-domain).

Q-Answer-timestamp Triple Creation.[5] Once we assembled the news corpus, we built (question, answer, timestamp) triples to accompany the new corpus as inputs for our task. To generate the needed triples we looked to crowdsourcing via Amazon Mechanical Turk. Our generation task consists of the following steps: (1) we selected a random news article from 2019 from the collected news corpus (these news articles are *gold articles* and will be hidden for experiments); (2) workers created questions, which *if posed before the respective article's publication date* would be seen as a forecasting question; (3) they indicated the answer, along with supporting evidence that the question consisted of (to ensure the correctness of the true answer); (4) they were asked to make multiple-choice distractors with their own knowledge and/or access to search engines; and (5) we ensured that a temporal phrase is present in the questions, for example: "*After May of 2020...*", "*... in June of 2021?*" to provide a temporal context (constraint) for each question, yielding more precise and well-defined forecasting questions. Completion of this task results in the desired triple of: a forecasting question, an answer to the question (with distractor choices), and a timestamp as our temporal constraint. The timestamp is set as the first day of the month in which the gold article was published.

To diversify questions in the dataset, we created two kinds of questions: binary yes-no questions and multiple-choice questions with *four choices*. Multiple-choice questions start with one of the six Ws (*i.e.*, who, what, when, where, why, and how) and are more challenging as they require determining the correctness of each choice.

Question Quality Verification. We performed a separate crowdsourcing data verification to test and enforce the following criteria: (1) is answering the question a *tractable* problem given (relevant)

[4] https://risk.lexisnexis.com

[5] Due to the limited space, for more details of our triple creation guidelines for human annotators, verification steps, and screenshots of our data collection/verification AMT interfaces, please refer to Sec. A of the appendix.

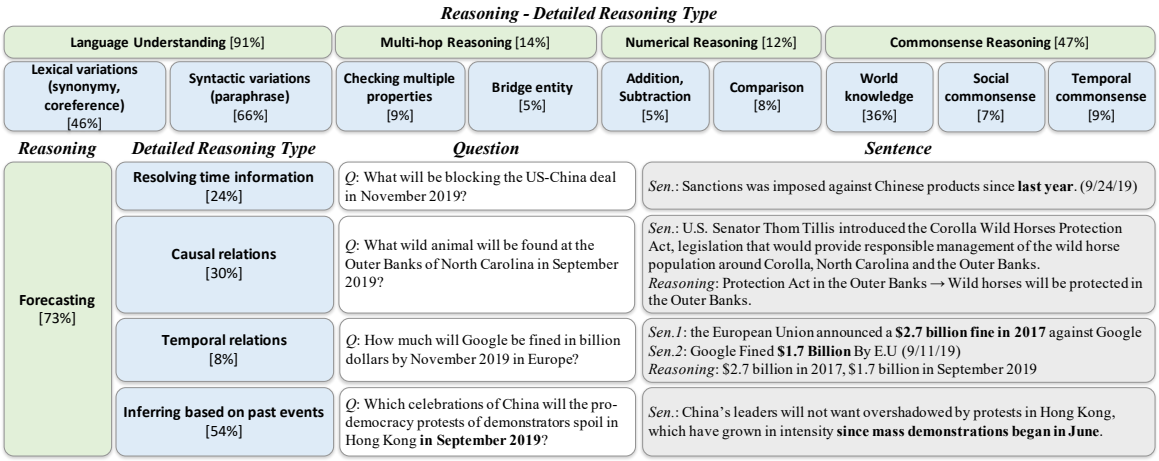

Figure 4: **Reasoning skills (types) and their frequency (in %) in the sampled data.** As each question can be labeled with multiple types, the total frequency does not sum to 100%. On average, 3 reasoning skills are required for each question. Examples of other reasoning types can be found in Fig. 11 in the appendix.

"past" articles?, and (2) is the question *deterministically* answerable given any article adhering to the question's temporal constraint? — If a question is too difficult, *i.e.*, an educated guess to the answer (when given relevant, constraint-adhering articles) is not possible, then we filter the question out. On the other hand, if the questions are answerable *with certainty* using "past" articles, or commonsense/world knowledge, then they are *not* considered to be forecasting questions. The desired response (majority vote from 3 annotators) is a "*yes*" for criterion (1) and "*no*" for (2), as that would show that the tuple of question and time constraint simulates the desired forecasting scenario. With the above method, we filtered out 31% of the questions collected in the triple creation step and were left with 5,704 yes-no questions and 4,513 multi-choice questions. More details about the verification step are included in Sec. A of the appendix.

4.2 Dataset Analysis

To better understand the properties of the questions in FORECASTQA, we examine: 1) a few data statistics 2) types of questions asked, and 3) the types of reasoning required to answer our questions.

Summary Statistics. FORECASTQA dataset is composed of 10,392 questions, divided into a 80/10/10 split of train, dev, and test data. Our 10k questions are roughly evenly split between multiple-choice and yes-no binary questions (Table 2). Over 17K distinct words were used to construct our questions and we have 218 unique time constraints associated with them; time constraints range from 2019-01-11 to 2019-11-12. We include additional statistics in Sec. D of appendix.

Types of Questions. To understand the types of questions in FORECASTQA, we examined the popular beginnings of sentences and created a tree-map plot (see Fig. 2). As shown, nearly half the questions start with the word *will* (44%), a result of over half of the questions being yes-no questions.

Reasoning Types. To examine types of reasoning required to answer our questions we sampled 100 questions and manually annotated them with reasoning types. Due to the forecasting nature of our dataset, we are particularly interested in questions containing the forecasting ability and thus spend more time looking into these questions. Our condensed results can be found in Figure 4, and more results from our cataloguing effort can be found in Sec. C of the appendix. Note that most questions contain more than one reasoning type.

5 Methods

To evaluate the forecasting capabilities of recent *multi-choice/binary* QA model architectures on FORECASTQA, we provide a comprehensive benchmarking analysis in this work. We run the experiments in two settings: (1) *closed-book* and (2) *constrained open-domain* setup. In the *closed-book* scenario only Q (question) and C (answer choices) are provided to the model (Q, C), while $\overline{A}$ (news articles) is provided for setting (2), $(Q, C, \overline{A})$[6]. We run these settings to understand the difficulty of both the closed-book and open-domain challenges presented by the questions in FORECASTQA.

[6] t_Q is always applied to $\overline{A}$, we left it out of the notation for simplicity.

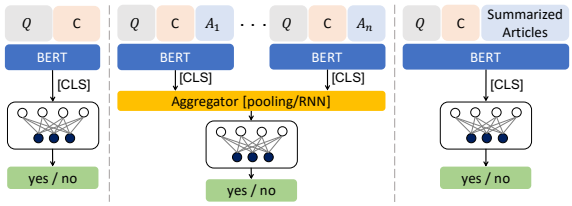

Figure 5: **Our baseline model architectures.** The CLS token is either fed into an MLP for classification or to the aggregator, which collects the information from each article before classifying.

For both settings, we explore several baseline models, but all follows a general architecture of a text encoder f and an optional context aggregation module g to aggregate information from a set of retrieved articles. Fig. 5 shows the architectures used. We model both yes-no and multiple-choice questions as a *binary* classification task; a model's prediction is the class with the largest probability. Below we introduce the details of our baselines.

Text Encoder. We use pre-trained language model, BERT (Devlin et al., 2019), as a text encoder (f from above)[7]. f is designed to deal with (Q, C) and $(Q, C, \overline{A})$ inputs, where $\overline{A}$ is a set of time-stamped articles that are retrieved from $\mathcal{A}$ to answer Q. Each input of f is transformed into $[\texttt{[CLS]}\,Q\,\texttt{[SEP]}\,C\,\texttt{[SEP]}\,A_i]$ (for each $A_i \in \overline{A}$, $C \in \mathcal{C}$), or $[\texttt{[CLS]}\,Q\,\texttt{[SEP]}\,C]$ (for each $C \in \mathcal{C}$) if articles are not supplied. The `[CLS]` token is the same as the one commonly used for fine-tuning PTLMs for a classification task, and `[SEP]` is the special separator token. The embedding of `[CLS]` is then used for predictions with an MLP layer (the leftmost model architecture in Fig. 5), or as input into a context aggregation module (the middle architecture in Fig. 5) subsequently introduced.

Context Aggregation (AGG). Two architectures are used when aggregating information from multiple, time-stamped articles $\overline{A}$ retrieved for a question. (1) *Temporal Aggregation:* This aggregator utilizes temporal ordering of the retrieved articles. Articles are sorted by their timestamps and their `[CLS]` token representation from f are aggregated by a Gated Recurrent Unit (GRU) (Cho et al., 2014) with a MLP head to make final predictions. (2) *Set Aggregation:* Alternatively, we ignore the temporal ordering of articles and use a *maxpooling operation* on the `[CLS]` token representations of each article. This pooled representation is passed to an MLP layer to make a prediction. Comparison between these aggregations helps understand the effect of modeling temporal order of evidence. These two aggregation modules are denoted by "AGG (GRU)" and "AGG (Maxpool)," respectively.

Multi-document Summarization (MDS). Rather than conducting context aggregation of the retrieved articles, we consider an MMR summarizer (Carbonell and Goldstein, 1998) which performs extractive, multi-document summarization of text to generate a summary A_{summ} (rightmost architecture in Fig. 5). The summary article A_{summ} is treated as if it is an $A_i \in \overline{A}$ and fed into a text encoder along with Q and C which then produce the `[CLS]` embedding for making a prediction. We name this method "MDS."

Integrated Approach. To take the best of both worlds in (Q, C) and $(Q, C, \overline{A})$ settings, we integrate two architectures (the leftmost and middle ones in Fig. 5). We concatenate the last two hidden representations of each architecture before passing the concatenated representation through a shared MLP layer. We use $\text{BERT}_{\text{LARGE}}$ as f in both architectures, AGG (GRU) for g and call this model "$\text{BERT}_{\text{LARGE}}$ ++ (integrated)" in Table 3.

Other Baselines. We also consider other baselines: ESIM (Chen et al., 2017b), BIDAF++ (Clark and Gardner, 2018), prepending extracted open event triples (Liu et al., 2019a) to BERT input, and a script learning approach, SAM-Net (Lv et al., 2019). We modify the approaches to fit into our setup. Detailed descriptions of each baseline method are included in Sec. E.3 of appendix.

6 Experiments

6.1 Experimental Setup

We adopt two types of settings: the closed-book setting (Q, C) and the constrained open-domain setting $(Q, C, \overline{A})$. In the constrained open-domain setting, we use BM25 (Robertson et al., 1995; Qi et al., 2019) as our IR method[8] to obtain $\overline{A}$, 10 retrieved articles. We also explore other IR methods in the later section. Note that we retrieve articles that do not violate the time constraints. We feed the question Q as a query and limit our access to articles in $\mathcal{A}$ by t_Q. Additionally, we validate the

[7] We did not include more recent pre-trained language models (*e.g.*, RoBERTa (Liu et al., 2019b), ALBERT (Lan et al., 2020), T5 (Raffel et al., 2020)) or pre-trained QA models like UnifiedQA (Khashabi et al., 2020), as these models are trained using text data published *after* the earliest timestamp in our dataset (2019-01-01), meaning information leakage could occur (and violates the forecasting setup). We tested more LMs in Sec. E.5 of appendix.

[8] Details of IR methods are described in appendix Sec. E.2.