A - **Parsing Header Information:**

1. For each document, get the header information within the SEC-XML tags `<SEC-HEADER>` and `</SEC-HEADER>`.
2. From the header, retrieve the following information:
   i) `COMPANY CONFORMED NAME`;
   ii) `CENTRAL INDEX KEY`;
   iii) `FORM TYPE`;
   iv) `CONFORMED PERIOD OF REPORT` (*optional*);
   v) `FILED AS OF DATE`;
3. The document naming scheme is: `<CENTRAL INDEX KEY>-<FILED AS OF DATE>.txt`.
4. If an item is successfully extracted from this document, write an entry into the comma-separated-value (CSV) file.
5. If not, the unparsed document is added to the log file.

B - **Parsing Text Information:**

1. Get the document text within the first SEC-XML tags `<TEXT>` and `</TEXT>`.
2. If the document contains HTML data, we need to remove any HTML tags. Before doing so, add line breaks before any `<P>` HTML tag to simplify paragraph detection in the next phase. Next ignore any HTML tags, and convert HTML entities in the form `&#...;` to their unicode value. Additionally, convert non-breaking spaces (U+00A0) to regular spaces, since non-breaking spaces are not considered whitespaces by the regular expressions.
3. Attempt to find an "Item 5" or, if none is found, an "Item 9":
   i) Match the text between the desired item title and the next item using tolerant regular expressions. More specifically, we identify a line starting with `ITEM 5 Operating` and a line starting with `ITEM 6`. For "Item 9", we identify a line starting with `ITEM 9 Management's Discussion` and a line starting with `ITEM 10`. In either case, we tolerate different capitalization, additional whitespaces, and formatting characters after the item number. [1]
   ii) If the matched text still contains the desired item title we ignore everything before the last item title which contains at least 100 words. This is mainly to avoid matching the table of content entry for the desired item.
4. If we were unable to match anything, or the matched text contains fewer than 100 words, then no text is extracted from this document.
5. If the item is successfully extracted, we remove additional markup:
   i) A page number followed by a `<PAGE>` tag and any whitespace before the page number, and after the tag. This is important for paragraph detection in the next phase.
   ii) Any table within `<TABLE>` and `</TABLE>` tags.
   iii) Any other remaining tags contained in angle brackets `<...>`.

C - **Extracting Paragraphs**

The data extracted from the last phase may still contain subtitles, page numbers, tables, and other content we wish to ignore. This phase cleans up the text by removing as much of this content as possible.

1. First, we split the text into paragraphs using empty lines which contain only whitespace to determine the seperation between one paragraph and the next.
2. Next we filter out the paragraphs which do not end with a punctuation: '.', ',', ':', ';', '!', '?'.

[1] The full regular expressions can be seen at: `https://github.com/mafaucher/SECir/blob/master/src/parser.py`.