Descriptions of Variation Attributes

Note: Each variation is a different context-question-historical answer combination. If a context has 8 questions, there will be 8 context-question combos, and each combo will be paired with as many historical answers there have been for that context paragraph. Additionally, context paragraphs that are longer than the max_seq_len will be split into overlapping sub-passages so that BERT can 1) use the whole passage and 2) have a sliding window of context

example of splitting text with sliding window/overlap:

- say we have sentence: 'The man went to the store to buy a gallon of milk' and we have a maximum sequence length of 5 and an overlap of 2
  - sub-passage/chunk 1: 'The man went to the'
  - sub-passage/chunk 2: 'to the store to buy'
  - sub-passage/chunk 3: 'to buy a gallon of'
  - sub-passage/chunk 4: 'gallon of milk'

- **all_features[n].unique_id:**
  - unique identifier of 1000000000 + n, where n in the index of this specific variation
  - `all_features[10].unique_id:`
    - `1000000010`
- **all_features[n].example_index:**
  - index of which example in examples this variation corresponds to; if a context paragraph is very long, it is divided into, for example, 2 overlapping sub-passage 'chunks' and therefore, every question for that context paragraph will be paired with each chunk; so, two variations in a row will have the same example_index (recall that each example is a different context paragraph-question combination!)
  - `all_features[10].example_index:`
    - `3`
      - this means that this variation corresponds to the third example
- **all_features[n].doc_span_index:**
  - indicates the index of the current 'chunk'; if a paragraph was split into sub-passages and we are looking at the first one, this number == 0
  - `all_features[10].doc_span_index:`
    - `0`
      - this means that this variation corresponds to the first chunk of the example's split-up context paragraph tokens, OR that a context paragraph was not split at all
- **all_features[n].tokens:**
  - BERT tokenizer output complete with '`[CLS]`' and '`[SEP]`' tokens
  - `all_features[10].tokens:`

```
['[CLS]', 'what', 'collaborations', 'did', 'she', 'do', 'with', 'nik', '##os', '?', '[SEP]',
'in', 'may', '1983', ',', 'she', 'married', 'nik', '##os', 'ka', '##r', '##vel', '##as', ',',
'a', 'composer', ',', 'with', 'whom', 'she', 'collaborated', 'in', '1975', 'and', 'in',
'november', 'she', 'gave', 'birth', 'to', 'her', 'daughter', 'sofia', '.', 'after', 'their',
'marriage', ',', 'she', 'started', 'a', 'close', 'collaboration', 'with', 'ka', '##r', '##vel',
'##as', '.', 'since', '1975', ',', 'all', 'her', 'releases', 'have', 'become', 'gold', 'or',
'platinum', 'and', 'have', 'included', 'songs', 'by', 'ka', '##r', '##vel', '##as', '.', 'in',
'1986', ',', 'she', 'participated', 'at', 'the', 'cypriot', 'national', 'final', 'for',
'eurovision', 'song', 'contest', 'with', 'the', 'song', 'the', '##lo', 'na', 'gin', '##o',
'star', '(', '"', 'i', 'want', 'to', 'be', 'a', 'star', '"', ')', ',', 'taking', 'second',
'place', '.', 'this', 'song', 'is', 'still', 'unreleased', 'up', 'to', 'date', '.', 'in',
'1984', ',', 'vis', '##si', 'left', 'her', 'record', 'company', 'emi', 'greece', 'and',
'signed', 'with', 'cbs', 'records', 'greece', ',', 'which', 'later', 'became', 'sony', 'music',
'greece', ',', 'a', 'collaboration', 'that', 'lasted', 'until', '2013', '.', 'in', 'march',
'1984', ',', 'she', 'released', 'na', "'", 'he', '##s', 'ka', '##rdi', '##a', '(', '"', 'if',
'you', 'had', 'a', 'heart', '"', ')', '.', 'the', 'album', 'was', 'certified', 'gold', '.',
'the', 'following', 'year', 'her', 'seventh', 'album', 'kat', '##i', 'sim', '##ven', '##i', '(',
'"', 'something', 'is', 'happening', '"', ')', 'was', 'released', 'which', 'included', 'one',
'of', 'her', 'most', 'famous', 'songs', ',', 'titled', '"', 'dod', '##eka', '"', '[', '"',
'twelve', '(', 'o', "'", 'clock', ')', '"', ']', 'and', 'reached', 'gold', 'status', 'selling',
'80', '.', '000', 'units', '.', 'in', '1986', 'i', 'ep', '##ome', '##ni', 'kin', '##isi', '(',
```

'"', 'the', 'next', 'move', '"', ')', 'was', 'released', '.', 'the', 'album', 'included', 'the', 'hit', 'pr', '##ag', '##mata', '(', '"', 'things', '"', ')', 'and', 'went', 'platinum', ',', 'becoming', 'the', 'best', 'selling', 'record', 'of', 'the', 'year', '.', 'in', 'february', '1988', 'she', 'released', 'her', 'ninth', 'album', 'tor', '##a', '(', '"', 'now', '"', ')', 'and', 'in', 'december', 'the', 'album', 'em', '##p', '##ne', '##fs', '##i', '!', '(', '"', 'inspiration', '!', '"', ')', 'which', 'went', 'gold', '.', 'in', '1988', ',', 'she', 'made', 'her', 'debut', 'as', 'a', 'radio', 'producer', 'on', 'ant', '##1', 'radio', '.', 'her', 'radio', 'program', 'was', 'titled', 'after', 'one', 'of', 'her', 'songs', 'ta', 'ko', '##rit', '##sia', 'ein', '##ai', 'ata', '##kt', '##a', '(', '"', 'girls', 'are', 'naughty', '"', ')', 'and', 'was', 'aired', 'every', 'weekend', '.', 'in', 'the', 'same', 'year', ',', 'she', 'participated', 'with', 'the', 'song', 'k', '##lai', '##o', '[SEP]']

- **all_features[n].token_to_orig_map:**
  - {tokenized_text_index: original_text_index}
  - `all_features[10].token_to_orig_map:`
    - {11: 0, 12: 1, 13: 2, 14: 2, 15: 3, 16: 4, 17: 5, 18: 5, 19: 6, 20: 6, 21: 6, 22: 6, 23: 6, 24: 7, 25: 8, 26: 8, 27: 9, 28: 10, 29: 11, 30: 12, 31: 13, 32: 14, 33: 15, 34: 16, 35: 17, 36: 18, 37: 19, 38: 20, 39: 21, 40: 22, 41: 23, 42: 24, 43: 24, 44: 25, 45: 26, 46: 27, 47: 27, 48: 28, 49: 29, 50: 30, 51: 31, 52: 32, 53: 33, 54: 34, 55: 34, 56: 34, 57: 34, 58: 34, 59: 35, 60: 36, 61: 36, 62: 37, 63: 38, 64: 39, 65: 40, 66: 41, 67: 42, 68: 43, 69: 44, 70: 45, 71: 46, 72: 47, 73: 48, 74: 49, 75: 50, 76: 50, 77: 50, 78: 50, 79: 50, 80: 51, 81: 52, 82: 52, 83: 53, 84: 54, 85: 55, 86: 56, 87: 57, 88: 58, 89: 59, 90: 60, 91: 61, 92: 62, 93: 63, 94: 64, 95: 65, 96: 66, 97: 67, 98: 67, 99: 68, 100: 69, 101: 69, ...}
      - we see above in the .tokens that the word at index 11 is the first word of the context paragraph as it comes immediately after the first '[SEP]' token, and we see that it corresponds to token 0 in the original text from the example, where the question and paragraph are NOT concatenated
- **all_features[n].is_max_context:**
  - {tokenized_text_index: True/False}
  - if True, this means that the token has the most context in this subpassage. Using example at the top of the page, 'buy' appears in two subpassages, 2 an 3. It has the most context in subpassage 3, though, as it has one word before and several words after, whereas in subpassage 2 it is the final word.
  - `all_features[10].is_max_context:`
    - {11: True, 12: True, 13: True, 14: True, 15: True, 16: True, 17: True, 18: True, 19: True, 20: True, 21: True, 22: True, 23: True, 24: True, 25: True, 26: True, 27: True, 28: True, 29: True, 30: True, 31: True, 32: True, 33: True, 34: True, 35: True, 36: True, 37: True, 38: True, 39: True, 40: True, 41: True, 42: True, 43: True, 44: True, ...}
- **all_features[n].input_ids:**
  - the ID that the BERT encoder uses to look ip the embedding for the word at that index; this is a list of numbers that will ALWAYS start with 101 (the ID for '[CLS]') and end with 102 (the ID for '[SEP]')
  - includes both question and passage
  - `all_features[10].input_ids:`
    - [101, 2054, 17437, 2106, 2016, 2079, 2007, 23205, 2891, 1029, 102, 1999, 2089, 3172, 1010, 2016, 2496, 23205, 2891, 10556, 2099, 15985, 3022, 1010, 1037, 4543, 1010, 2007, 3183, 2016, 8678, 1999, 3339, 1998, 1999, 2281, 2016, 2435, 4182, 2000, 2014, 2684, 8755, 1012, 2044, 2037, 3510, 1010, 2016, 2318, 1037, 2485, 5792, 2007, 10556, 2099, 15985, 3022, 1012, 2144, 3339, 1010, 2035, 2014, 7085, 2031, 2468, 2751, 2030, 8899, 1998, 2031, 2443, 2774, 2011, 10556, 2099, 15985, 3022, 1012, 1999, 3069, 1010, 2016, 4194, 2012, 1996, 18543, 2120, 2345, 2005, 12714, 2299, 5049, 2007, 1996, 2299, 1996, 4135, 6583, 18353, 2080, 2732, 1006, 1000, 1045, 2215, 2000, 2022, 1037, 2732, 1000, 1007, 1010, 2635, 2117, 2173, 1012, 2023, 2299, 2003, 2145, 13270, 2039, 2000, 3058, 1012, 1999, 3118, 1010, 25292, 5332, 2187, 2014, 2501, 2194, 12495, 5483, 1998, 2772, 2007, 6568, 2636, 5483, 1010, 2029, 2101, 2150, 8412, 2189, 5483, 1010, 1037, 5792, 2008, 6354, 2127, 2286, 1012, 1999, 2233, 3118, 1010, 2016, 2207, 6583, 1005, 2002, 2015, 10556, 17080, 2050, 1006, 1000, 2065, 2017, 2018, 1037, 2540, 1000, 1007, 1012, 1996, 2201, 2001, 7378, 2751, 1012, 1996, 2206, 2095, 2014, 5066, 2201, 10645, 2072, 21934, 8159, 2072, 1006, 1000, 2242, 2003, 6230, 1000, 1007, 2001, 2207, 2029, 2443, 2028, 1997, 2014, 2087, 3297, 2774, 1010, 4159, 1000, 26489, 19025, 1000, 1031, 1000, 4376, 1006, 1051, 1005, 5119, 1007, 1000, 1033, 1998, 2584, 2751, 3570, 4855, 3770, 1012, 2199, 3197, 1012, 1999, 3069, 1045, 4958, 8462, 3490, 12631, 17417, 1006, 1000, 1996, 2279, 2693, 1000, 1007, 2001, 2207, 1012, 1996, 2201, 2443, 1996, 2718, 10975, 8490, 21022, 1006, 1000, 2477, 1000, 1007, 1998, 2253, 8899, 1010, 3352, 1996, 2190, 4855, 2501, 1997, 1996, 2095, 1012, 1999, 2337, 2997, 2016, 2207, 2014, 6619, 2201, 17153, 2050, 1006, 1000, 2085, 1000, 1007, 1998, 1999, 2285, 1996, 2201, 7861, 2361, 2638, 10343, 2072, 999, 1006, 1000, 7780, 999, 1000, 1007, 2029, 2253, 2751, 1012, 1999, 2997, 1010, 2016, 2081, 2014, 2834, 2004, 1037, 2557, 3135, 2006, 14405, 2487, 2557, 1012, 2014, 2557, 2565, 2001, 4159, 2044, 2028, 1997, 2014, 2774, 11937, 12849, 14778, 8464, 16417, 4886, 29533, 25509, 2050, 1006, 1000, 3057, 2024, 20355, 1000, 1007, 1998, 2001, 4836, 2296, 5353, 1012, 1999, 1996, 2168, 2095, 1010, 2016, 4194, 2007, 1996, 2299, 1047, 19771, 2080, 102]
- **all_features[n].input_mask:**

- list/array of 0 and 1 where 1 at a given index means that the corresponding token at that same index is an actual token and 0 means that the token at that index is just padding
- there will only be zeroes if the context paragraph length is less than the maximum seq length or if the context paragraph was split into sub passages and the final subpassage length is less than the maximum sequence length
- includes both question and passage
- `all_features[10].input_mask:`
  - ```
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    ```

- **all_features[n].segment_ids:**
  - list/array of 0 and 1 where 0 at a given index means that the corresponding token at that same index is part of the question text (between '[CLS]' token and first '[SEP]' and includes those tokens) OR padding and 1 means that the token at that index is part of the context paragraph (starting with the first token after the first '[SEP]' and including '[SEP]' token at the end)
  - there will only be zeroes if the context paragraph length is less than the maximum seq length or if the context paragraph was split into sub passages and the final subpassage length is less than the maximum sequence length
  - includes both question and passage
  - `all_features[10].segment_ids:`
    - ```
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
      ```
- **all_features[n].start_position:**
  - indicates token index of beginning of answer span; will be negative if answer is contained within previous subpassage **or** larger than maximum sequence length/length of current subpassage if answer is in the next subpassage
  - `all_features[10].start_position:`
    - `59`
- **all_features[n].end_position:**
  - same as start_position except indicates the last word in an answer span
  - `all_features[10].end_position:`
    - `79`
- **all_features[n].history_answer_marker:**
  - 0 where token not in answer history, 1 where it is
  - only indicates the answer for 1 history at a time/1 turn of conversation history
  - therefore, all information above will be repeated n times for n answer histories, with the only difference being the history_answer_marker and the metadata (next)
  - `all_features[10].history_answer_marker:`
    - ```
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      ```
      ```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- **all_features[n].metadata:** dictionary of metadata
  - **'turn':**
    - starts at '1' and indicates whether the current question (in .tokens) is the first, third, eighth, etc question for the current paragraph
    - `all_features[10].metadata['turn']`
      - `"turn": 4`
  - **'history_turns':**
    - array containing single element to specify which history turn is specified in history_answer_marker and in metadata
    - `all_features[10].metadata['history_turns']`
      - `"turn": 2`
  - **'tok_history_answer_markers':**
    - array/list containing 1 array/list; when counting up words split at only whitespace in the original, untokenized context paragraph from examples, 1 indicates the answer tokens for current history turning specified in 'history_turns' in that piece of text; 0 means that token in not an answer for the current history question
    - `all_features[10].metadata['tok_history_answer_markers']`

      -
        ```
        [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0]]
        ```
  - **'followup':**
    - 'y', 'n', or 'm'
    - whether or not there should, shouldn't, or could be a follow-up question asked after this question
    - `all_features[10].metadata['followup']`
      - `"followup": "m"`
  - **'yesno':**
    - 'y' or 'x'
    - whether the current question is a yes or no question
    - `all_features[10].metadata['yesno']`
      - `"yesno": "x"`
  - **'history_turns_text':**
    - list of lists of strings; historical questions and their answers
    - `all_features[10].metadata['history_turns_text']`
      - `'history_turns_text': [('did they have any children?', 'in November she gave birth to her daughter Sofia.')]`