August 2022

Automatic Detection of Phrase Boundaries and Highlighting Phrases in Text

Ajit Narayanan

Rain Breaw Michaels

Automatic Detection of Phrase Boundaries and Highlighting Phrases in Text

ABSTRACT

This disclosure describes techniques to display text in a way that enhances readability through phrasal grouping. Using dependency-parsing techniques, sentences are automatically split into phrases. The display of text is adapted to phrasal groupings, e.g., using highlighting, with or without synchronized audio. For example, a text-to-speech (TTS) reader voices phrases sequentially while the phrase currently being voiced is synchronously highlighted. The grouping of text into phrases and their highlighting during audio-visual consumption by the user can improve comprehension.

KEYWORDS

- Phrase boundary
- Sentence boundary
- Phrase highlighting
- Phrasal grouping
- Scooping
- Dependency tree
- Dependency parsing
- Noun-phrase chunking
- Text-to-speech
- Phrase chunking
- Natural language processing (NLP)
- Saccades

BACKGROUND

The consumption of textual information is a significant form of digital interaction. The reading of text is sequential, working through the optico-neural system. However, the brain does not process individual characters sequentially; instead, the eye moves in discrete intervals between small groupings of words at a time, in a process known as saccading. There is empirical evidence that the brain reads one phrase at a time, mirroring natural languages, which have a syntactic structure that creates larger and larger chunks of meaning through phrases.

Language is both lexical and paradigmatic, e.g., the meaning of a word is impacted by the grammar and other words next to it. For example, the word “like” has different meanings in the phrases, “I, like you” (similar to) and “I like you” (enjoy).

The user interface of nearly all digital reading surfaces display text conventionally, e.g., as though the text was displayed on non-interactive media like paper, thereby missing an opportunity for enhanced comprehension through aligning phrase structure with user interface.

Where interactivity with digital text has been implemented, it combines visual display with a read-aloud capability, e.g., text-to-speech (TTS). The consumption of text in audio *and* visual formats enhances comprehension over both audio-only and visual-only text modes. Comprehension can be further improved by highlighting the section of text currently being read out. The highlight is commonly at the level of a word, a sentence, or a paragraph, and it moves with the audio read-out of the text. Highlighting text keeps the user’s visual attention in the same area that the user’s auditory attention is, enhancing comprehension and engagement. While empirical evidence indicates that the brain reads text in phrases, currently, highlighting isn't done at phrase level.

To the extent phrase-based comprehension enhancement exists today, it is a highly manual and tedious process. For example, motivated by research that shows phrase-based reading to have superior outcomes, educators use tools [8] [9] to manually and painstakingly create phrases for early readers, a task known as scooping.

DESCRIPTION

This disclosure describes techniques to display text in a way that enhances readability through phrasal grouping. Phrases are determined by automatically determining phrase boundaries in the text. The display of text is adapted to phrasal groupings (or chunkings), e.g.,

using highlighting, with or without synchronized audio. The grouping of text into phrases and their highlighting improves comprehension. Phrasal highlighting can be incorporated into a text read-aloud / TTS experience.

Detection of phrase boundaries

Phrase chunking accepts a sentence as input, performs a dependency parse of the sentence to create a dependency tree, and produces phrase boundaries as output.

Fig. 1: Dependency tree for a sentence written in Cyrillic script

Dependency parsing (and the resulting creation of a dependency tree) is a well-understood natural-language processing (NLP) task [13] and can be done in most natural languages. Fig. 1 illustrates an example of dependency parsing in Cyrillic script. Phrase cues can be automatically created for an article or long-form text as follows:

- Use sentence and word chunking to divide the article into sentences.
- For each sentence, use a dependency-parsing algorithm to create a dependency tree.
- Use the dependency tree to identify phrase boundaries, optimizing for particular applications or user interfaces.

For example, Fig. 2 illustrates a sentence (in box) and a dependency tree for the sentence, created using standard dependency-parsing techniques.

Please create a document with questions about our preferences for communication, and fill it out after the meeting.

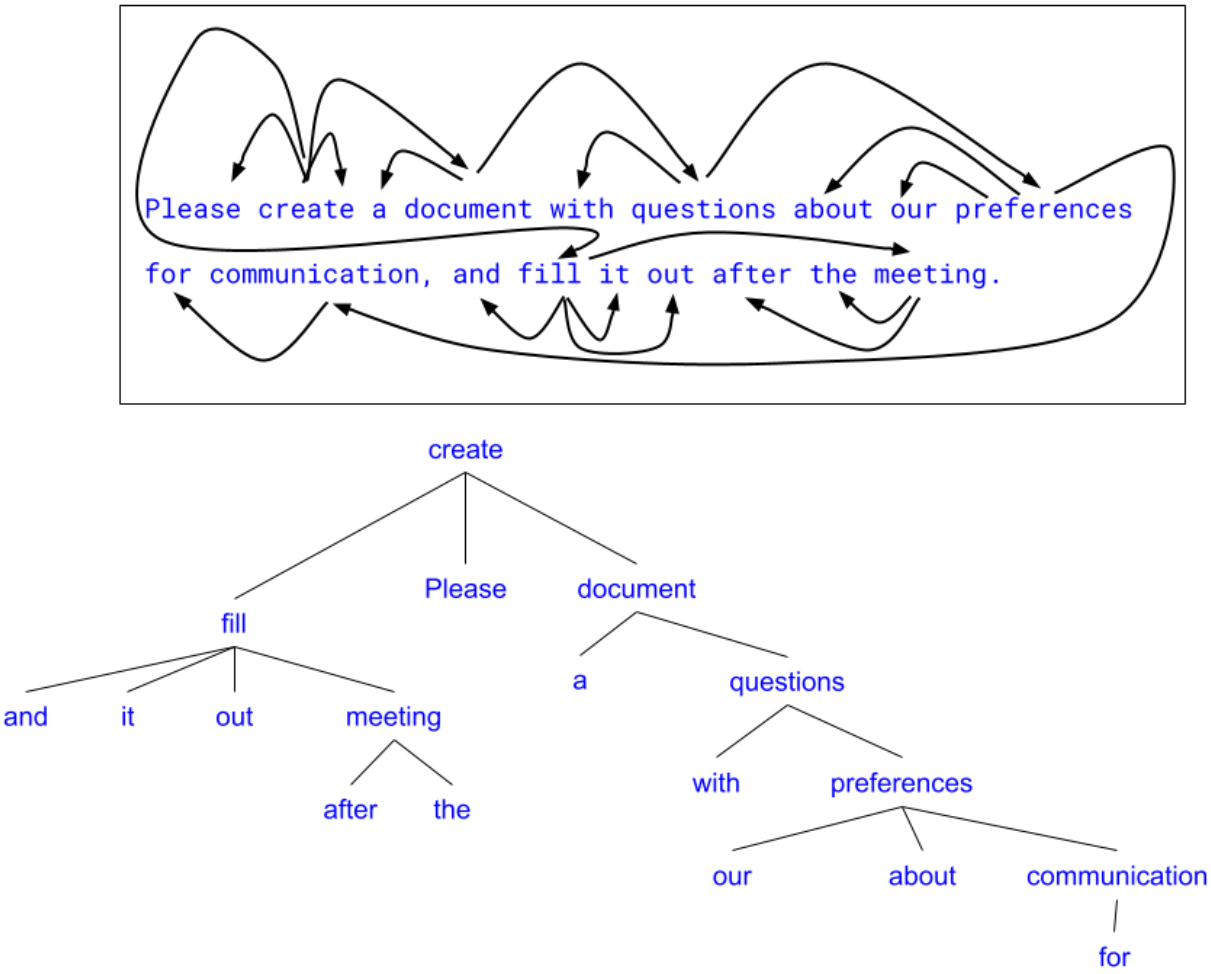

Fig. 2: An example sentence and corresponding dependency tree

Each edge in the dependency tree is labeled with a weight that equals the distance (in words) between the words that the edge connects. For example, the edge that connects the words ‘create’ and ‘document’ has a weight of 2, since it connects words that are two words apart. The edge that connects ‘create’ and ‘fill’ is 11, as it connects words that are 11 words apart. The resulting edge-weighted tree is illustrated in Fig. 3.

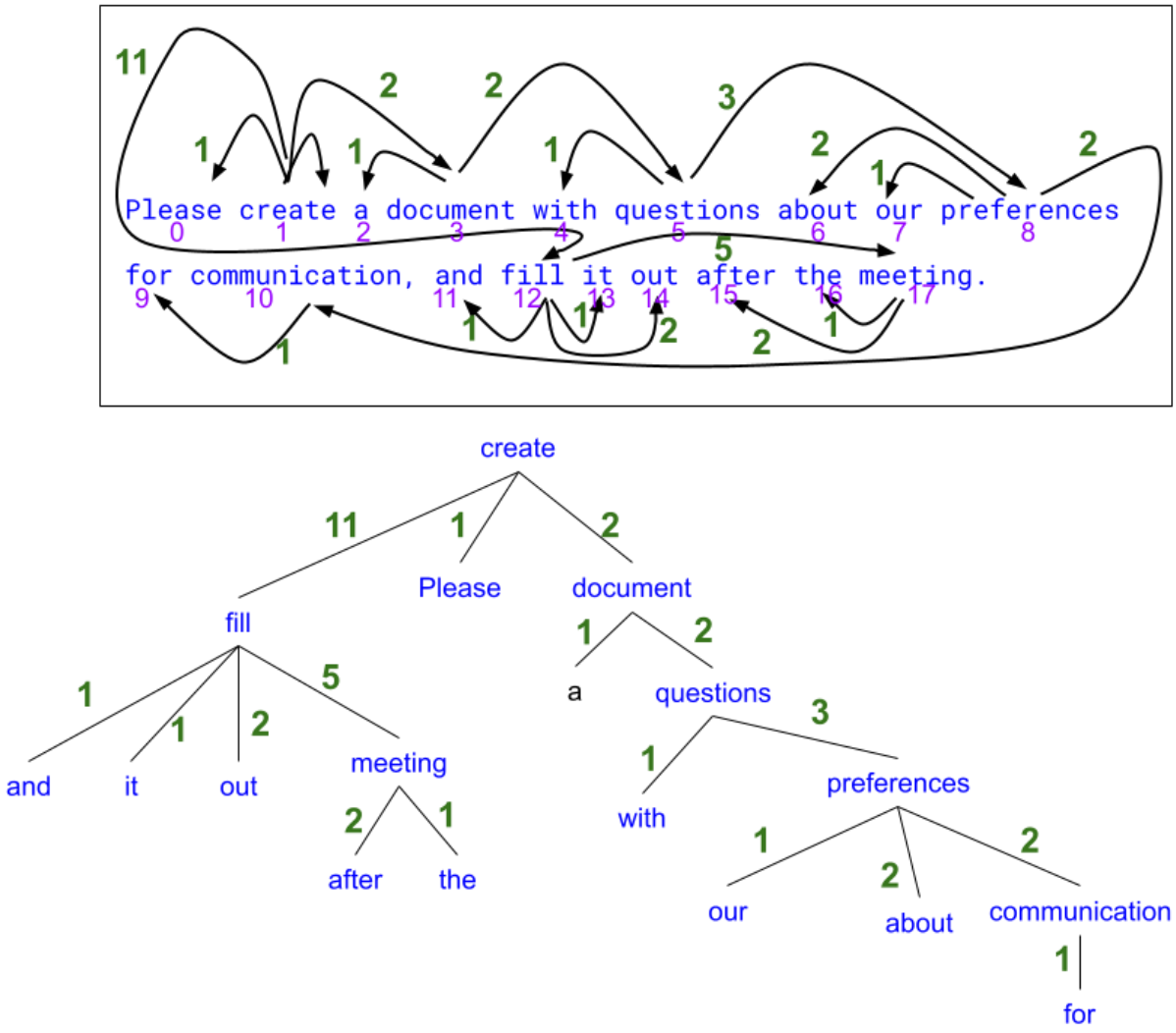

Fig. 3: Dependency tree with edges labeled with the distance (in words) between nodes

In Fig. 3, the edge labels, which indicate distance (in words) between nodes, are shown in green. The purple numbers are indices of each word (starting with 0 for ‘please’).

A predicate is created that enables the definition of an allowable phrase for the purposes of a particular application (e.g., highlighting) or user interface. An example predicate is that the number of words in a phrase be less than or equal to 3. This results in a splitting of the dependency tree, as illustrated in Fig. 4. The red numbers next to the red crosses represent the sequence in which the edges are removed, per an example procedure illustrated further below in Fig. 5.

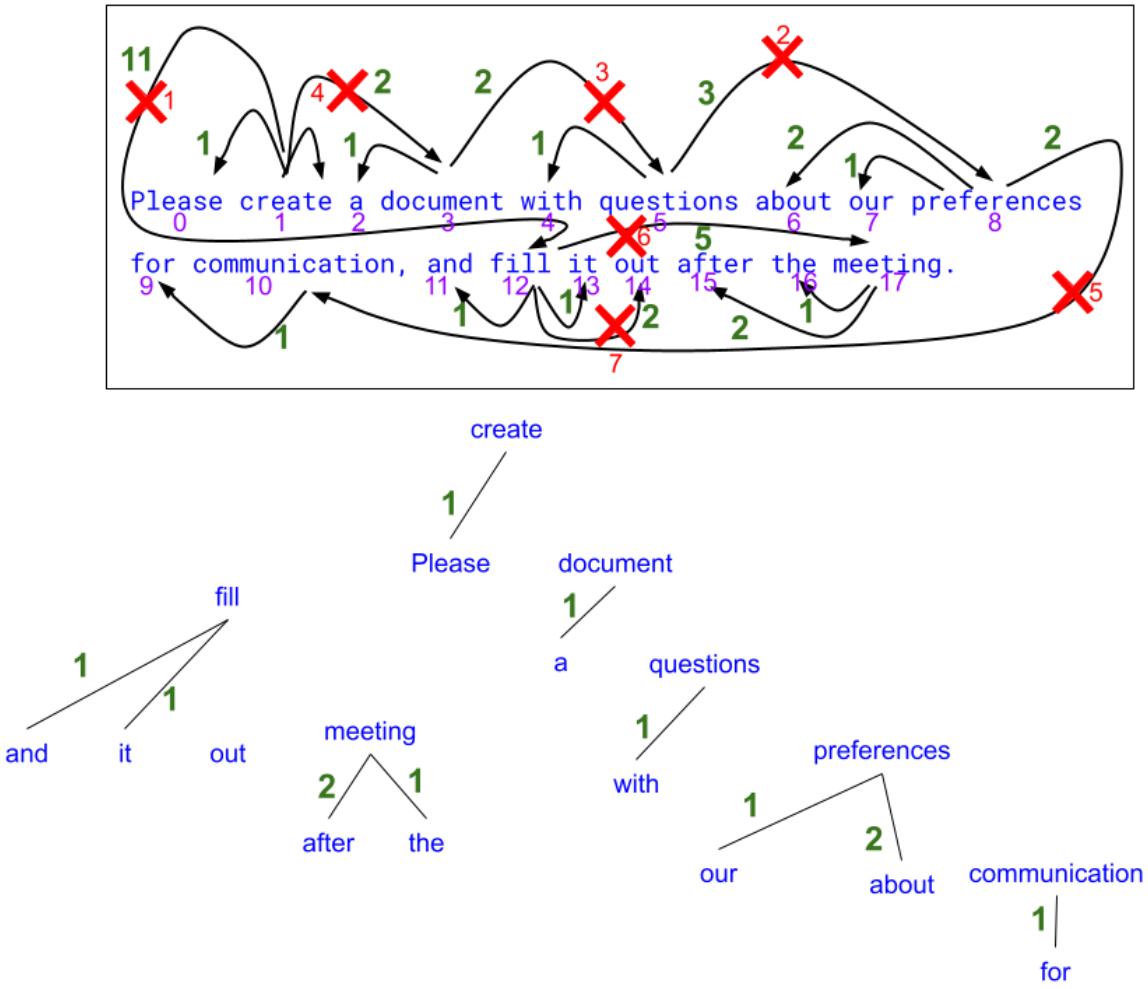

Fig. 4: Setting a predicate that phrase length be less than or equal to three splits the dependency tree

The phrase grouping that results from the dependency-tree split is shown below (phrase boundaries are marked with a single slash, and sentence boundaries are marked with two slashes).

> Please create / a document / with questions / about our preferences / for communication, / and fill it / out / after the meeting.//

In this manner, a sentence can be split into phrases, each of which can be highlighted alongside synchronized text-to-speech for enhanced reading comprehension.

```
def split_graph(graph, predicate):
 subgraphs = []
 def _split_graph(graph):
   if predicate(graph):
     subgraphs.append(graph)
     return
   all_edges = [(wt, u, v) for (u, v, wt) in graph.edges.data('weight')]
   all_edges.sort(reverse=True)
   max_edge = all_edges[0]
   graph.remove_edge(max_edge[1], max_edge[2])
   for c in connected_components(graph):
     _split_graph(graph.subgraph(c).copy())
 _split_graph(graph)
 return subgraphs
```

Fig. 5: Pseudocode to split a dependency tree given a predicate

Fig. 5 illustrates an example of pseudocode to split the dependency tree given a predicate. As illustrated, the procedure `_split_graph` is recursively called.

- Each recursive call first checks if the tree that is passed in as its argument matches the predicate.
- If it matches the predicate, the tree is considered a qualifying phrase, and no further recursion is done on it.
- If it does not match the predicate,
    a. choose the edge that has the largest weight and remove it from the tree, resulting in the tree separating into two connected components.
    b. `_split_graph` is called on each of the connected components.
- The procedure terminates when the tree has been split into a number of connected components, where each connected component is a phrase that matches the predicate.

A procedure similar to the one illustrated in Fig. 5 can be used for other predicates or use cases, which result in differing edge labels and tree-splits. For example, if a predicate is to restrict phrases to a certain number, e.g., twenty, of characters, the tree-split of Fig. 6 results.

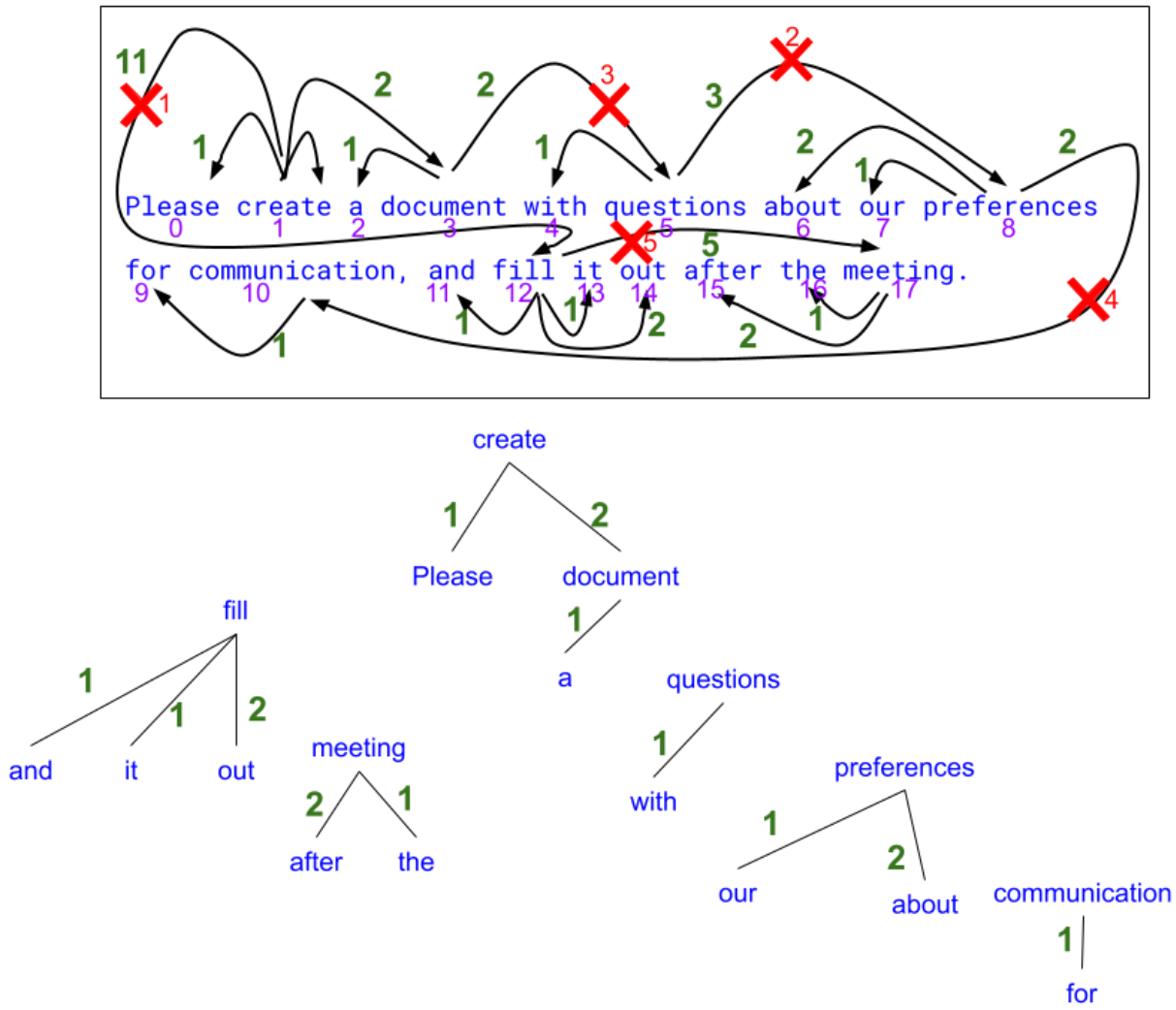

Fig. 6: Setting a predicate that phrase length be twenty or fewer characters splits the dependency tree

The phrase boundaries corresponding to the disjoint tree illustrated in Fig. 6 is as below.

Please create a document / with questions / about our preferences / for communication, / and fill it out / after the meeting.//

Phrase boundaries are marked with a single slash, and sentence boundaries are marked with two slashes. It is seen that each phrase has twenty or fewer characters, as predicated.

Alternatively, edge weights can be determined using dependency labels and edge directionality, such that some kinds of phrases are given more prominence than others (for example, always cutting coordinating conjunctions). More direct visual predicates can also be used: for example, whether a phrase, when rendered in a particular font, fits within 50 pixels. A procedure similar to the one illustrated in Fig. 5 can be used to determine phrase boundaries based on given predicates.

Using phrase boundaries for enhanced reading comprehension

Phrase boundaries can be used to enhance reading comprehension in various ways, e.g., phrase-cued text-to-speech; phrase-based highlighting synchronous with eye gaze; phrase breaks for displaying short columns; phrase-aligned line breaks; phrase-based text justification; etc. Some phrase-cued applications are explained in greater detail below.

Phrase-cued highlighting for text-to-speech

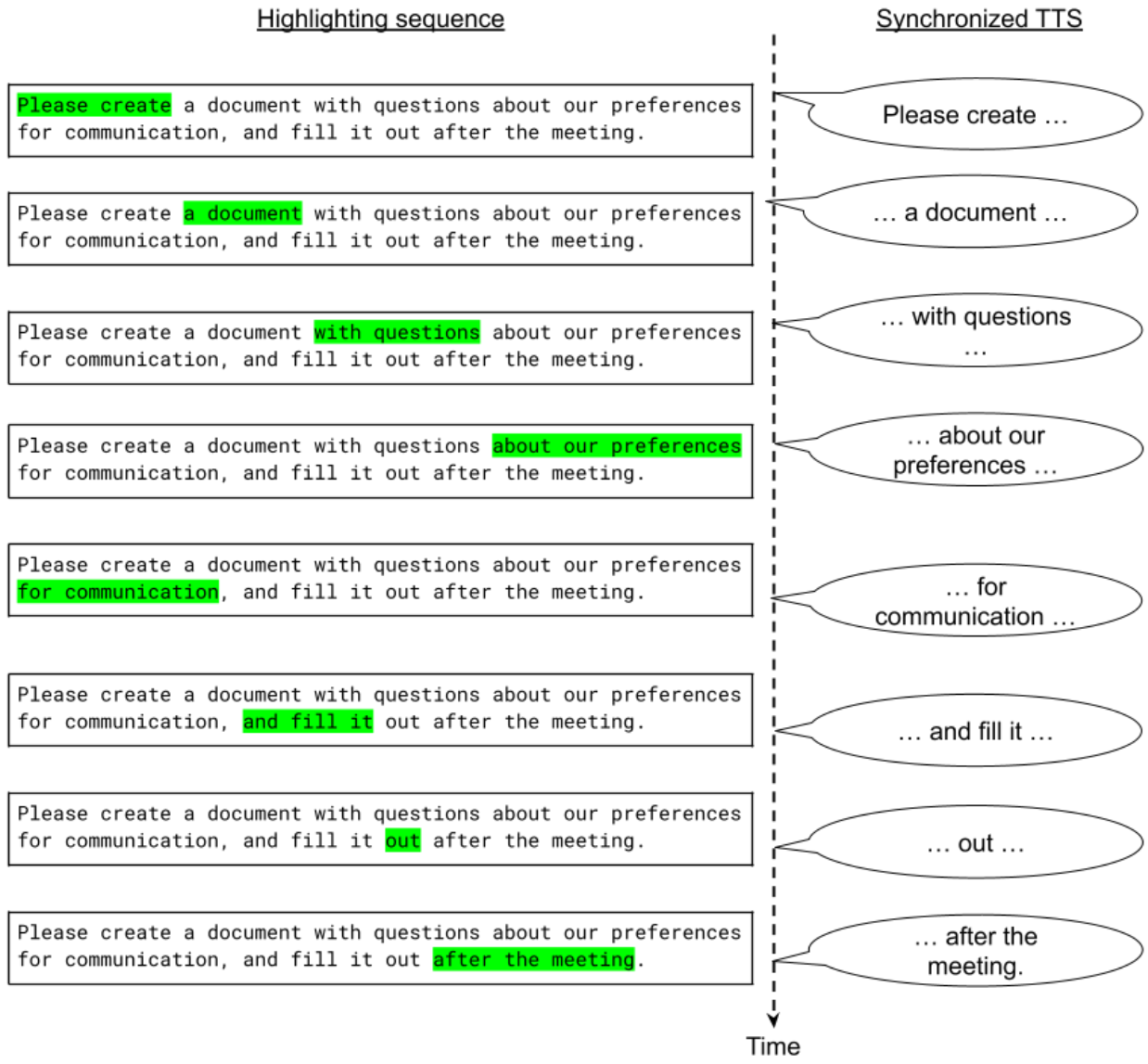

Fig. 7: Phrase-cued highlighting for TTS

Phrase boundaries obtained using the above-described procedures can be used to enhance reading comprehension, for example, using phrase-cued highlighting for TTS illustrated in Fig. 7. A text-to-speech (TTS) reader groups text into phrases and voices phrases sequentially. The phrase currently being voiced is synchronously highlighted. This is in contrast to traditional reading comprehension techniques, which highlight words, sentences, or paragraphs, none of which correspond naturally to the way the human optico-neural system functions.

Phrase-cued highlighting for eye-gaze

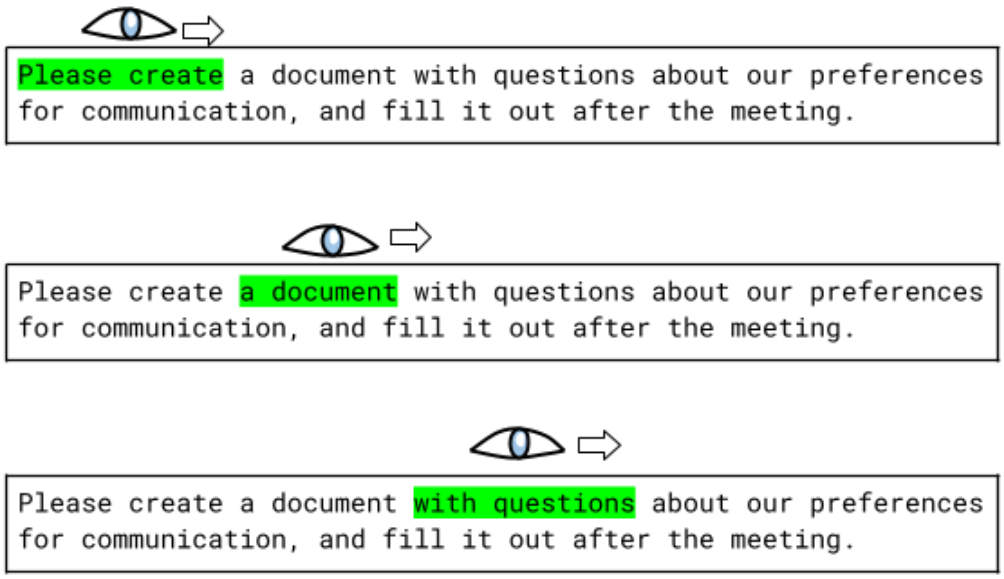

Fig. 8: Phrase-cued highlighting for eye-gaze

Illustrated in Fig. 8, phrase-highlighting can also follow the eye gaze of the user. An eye-gaze detector identifies the current location of the user's eye fixation, and the corresponding phrase is highlighted. Using phrase cues for highlighting the location of the user's eye gaze enables the user to visualize words that are closely related syntactically.

Phrase breaks for displaying short columns

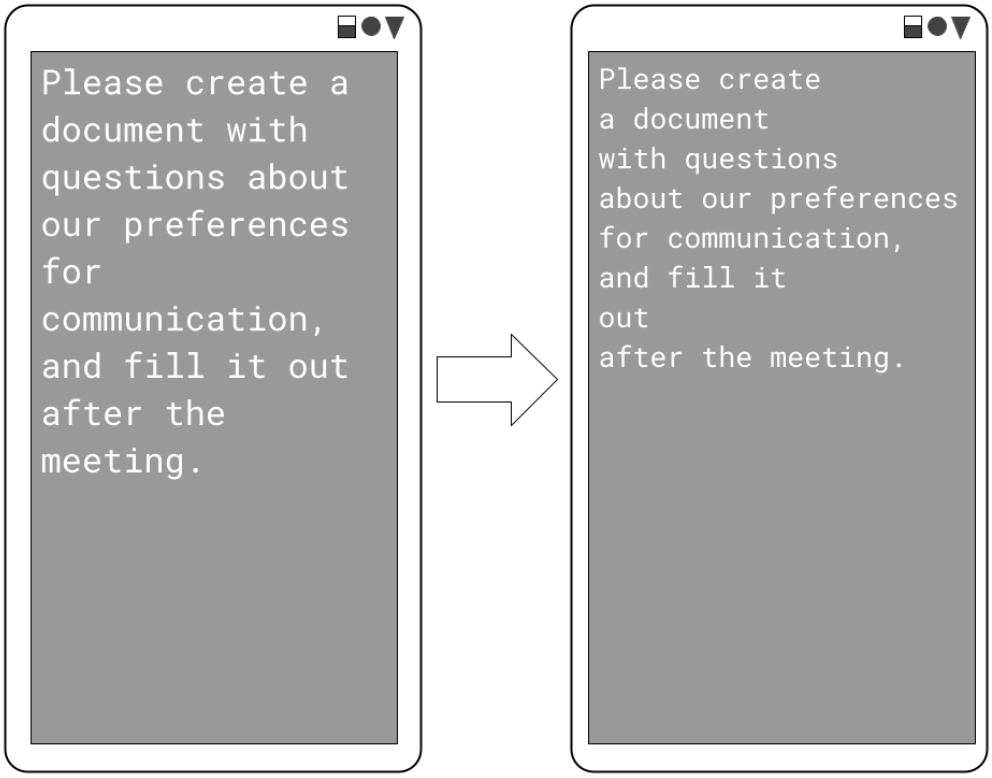

Fig. 9: Phrase breaks for displaying short columns (a) Traditional column display, which doesn't account for the phrase structure of sentences (b); Phrase-based column display, which reduces eye-saccading

Illustrated in Fig. 9, phrase breaks can be used to display short columns (as in a smartphone), reducing eye-saccading by adjusting the column width to match phrase lengths. Alternatively, line breaks can be inserted into the text such that they fit a predetermined column width, governed by, for example, the width of the screen on which the text is rendered. Fig. 9(b) illustrates column display based on a monospaced font and a three-word (or less) per-phrase predicate. The techniques apply to variable-width fonts as well as other phrase-splitting predicates, e.g., characters-per-phrase, the space taken by the text when rendered in a particular font, etc.

Phrase-aligned line breaks

A phrase split across lines causes readers to swing their eyesight between opposite edges of the screen to complete the comprehension of the phrase. This misalignment of line breaks with phrase breaks can be disorienting, an effect accentuated by lengthy sentences.

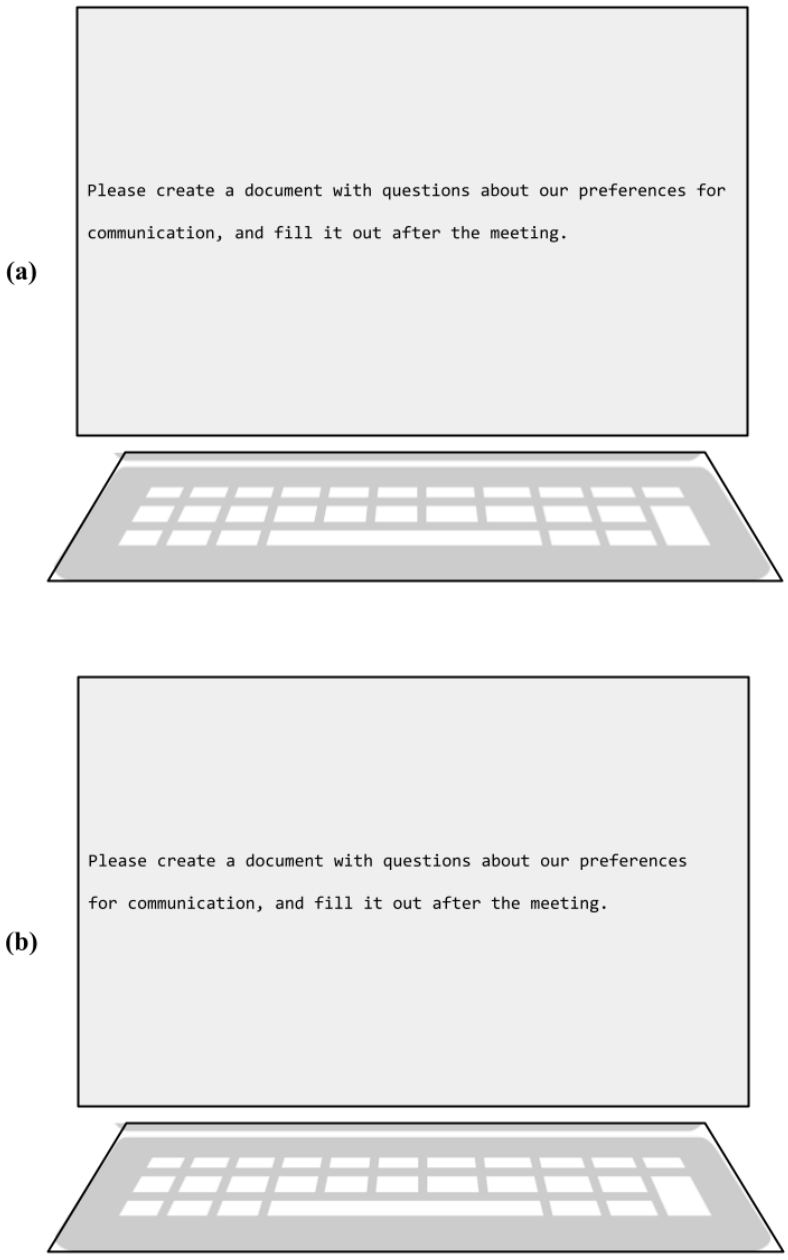

Fig. 10: Line breaks in text (a) Traditional line-breaking, incognizant of phrase structure, which results in disorientation and visual discomfort; (b) Phrase-based line-breaking, which reduces visual discomfort.

As illustrated in the example of Fig 10(a) (traditional line breaking) the line is broken after the word ‘for,’ such that the phrase ‘for communication’ is split between two lines, leading to disorientation and visual discomfort. Fig. 10(b) illustrates line-breaking cognizant of phrase structure, such that line breaks don’t split phases, thereby reducing visual discomfort and increasing comprehensibility. Generally, the techniques of phrase-cognizant line breaking can be used to perform text justification for an entire paragraph or article, breaking lines to optimize comprehension. A dynamic programming algorithm can optimize overall phrase breaking by keeping track of the cognitive cost of breaking lines at different locations.

In this manner, the described techniques improve the experience of reading text for users across virtually any digital reading surface such as laptops, tablets, smartphones, etc. Some example phrase-cued applications include phrase-cued text-to-speech, phrase-based highlighting synchronous with eye gaze, phrase breaks for displaying short columns, phrase-aligned line breaks, phrase-based text justification, etc. Enhancing users’ abilities to comprehend text can positively impact their opportunities, productivity and well-being.

CONCLUSION

This disclosure describes techniques to display text in a way that enhances readability through phrasal grouping. Using dependency-parsing techniques, sentences are automatically split into phrases. The display of text is adapted to phrasal groupings, e.g., using highlighting, with or without synchronized audio. For example, a text-to-speech (TTS) reader voices phrases sequentially while the phrase currently being voiced is synchronously highlighted. The grouping of text into phrases and their highlighting during audio-visual consumption by the user can improve comprehension.

REFERENCES

[1] Lv, Xin, Tiejun Zhao, Zhan-yi Liu, and Muyun Yang. “Automatic detection of prosody phrase boundaries for text-to-speech system.” In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pp. 135-141. 2001.

[2] http://www.readsy.co/ accessed 13 Jul 2022.

[3] “Read faster and easier, all day long” https://www.beelinereader.com/ accessed 13 Jul 2022.

[4] “Digital learning tools from Microsoft education” https://www.microsoft.com/en-us/education/products/learning-tools accessed 13 Jul 2022.

[5] “Select to speak (for developers)” https://chromium.googlesource.com/chromium/src/+/HEAD/docs/accessibility/os/select_to_speak.md accessed 13 Jul 2022.

[6] https://www.audible.com/ep/wfs accessed 13 Jul 2022.

[7] “The Economist in audio” https://www.economist.com/audio-edition accessed 13 Jul 2022.

[8] “Phrase-cued text lessons” https://www.interventioncentral.org/academic-interventions/reading-comprehension/phrase-cued-text-lessons accessed 13 Jul 2022.

[9] “Phrase cued text generator” https://www.interventioncentral.org/rti2/phrase_cues accessed 13 Jul 2022.

[10] “Sentence boundary disambiguation.” https://en.wikipedia.org/wiki/Sentence_boundary_disambiguation accessed 13 Jul 2022.

[11] “Extracting information from text” https://www.nltk.org/book/ch07.html accessed 13 Jul 2022.

[12] Akbik, Alan, Duncan Blythe, and Roland Vollgraf. "Contextual string embeddings for sequence labeling." In *Proceedings of the 27th international conference on computational linguistics*, pp. 1638-1649. 2018. https://aclanthology.org/C18-1139/

[13] Weiss, David and Petrov, Slav. “An upgrade to Syntaxnet, new models and a parsing competition.” https://ai.googleblog.com/2017/03/an-upgrade-to-syntaxnet-new-models-and.html

[14] “Component (graph theory)” https://en.wikipedia.org/wiki/Component_(graph_theory) accessed 13 Jul 2022.

[15] Nomvete, Patsy, and Susan R. Easterbrooks. “Phrase-reading mediates between words and syntax in struggling adolescent readers.” *Communication Disorders Quarterly* 41, no. 3 (2020): 162-175. https://journals.sagepub.com/doi/abs/10.1177/1525740119825616