1. Parsing and lemmatization

a) Lemmatization is important in NLP because it allows different word forms to be treated as a single unit of meaning, which is useful for tasks like search engines, keyword extraction, and word frequency analysis (Slide: *Lemmatization – What's the point?*). By reducing words to their base form, systems can better generalize and improve accuracy. However, some applications require inflected forms to function correctly. For example, search engines benefit the most from using lemmas, as users searching for "running" should also find results for "run" and "ran." On the other hand, machine translation does not work properly if only lemmas are used because inflectional differences carry important grammatical meaning that needs to be preserved (Slide: *What's hard about lemmatizing?*). Therefore, lemmatization is valuable in simplifying text but should be applied selectively based on the task.

b) Parsing is very important in systems that need to understand the structure of sentences. This includes machine translation, grammar correction, and question answering systems because they rely on knowing how words fit together in a sentence. For example, in machine translation, a sentence like *"The dog chased the cat"* could be wrongly translated if the system doesn't recognize that "the dog" is the subject and "the cat" is the object. If parsed incorrectly, it might translate as if *the cat* was chasing *the dog which* completely changes the meaning (Slide: *Syntactic parsing (constituency parsing)*). Also in error correction, knowing the sentence structure helps identify mistakes, such as missing words or incorrect verb forms. Without proper parsing these systems would struggle to provide accurate results.

c) When designing systems the decision to train on lemmatized data depends on the specific requirements of each task.

   Sentiment Analysis: Lemmatization can be beneficial here, as it reduces words to their base forms, allowing the model to recognize different inflections of a word as conveying the same sentiment. For example, "happy" and "happiness" would both be reduced to "happy," aiding in consistent sentiment detection (Suvrat, 2025).

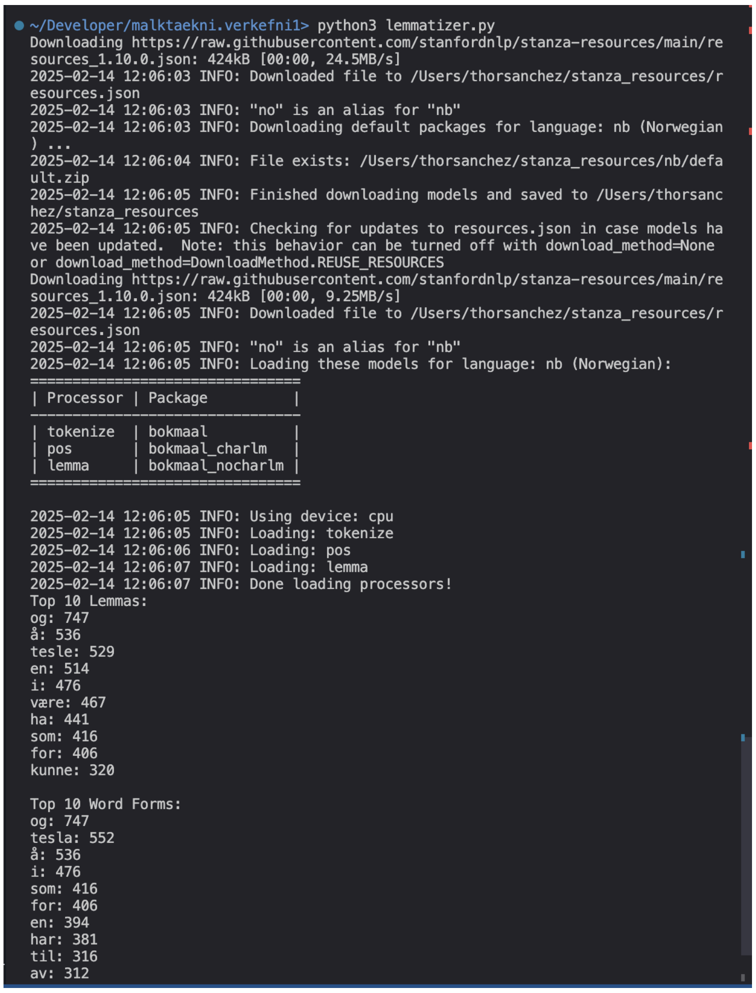
Keyword Extraction: Applying lemmatization helps in identifying the core meaning of words, ensuring that variations like “run” and “running” are treated as the same keyword. This consolidation improves the accuracy of extracted keywords by focusing on their root forms (Chandrakant, 2024).

Automatic Paraphrasing: Here preserving the original inflected forms is crucial to maintain the grammatical structure and nuance of the text. Training on lemmatized data might strip away necessary context which leads to paraphrases that are grammatically incorrect or that fail to capture the intended meaning (Roe, 2022).

In summary, while lemmatization aids in standardizing input data for sentiment analysis and keyword extraction, it is generally avoided in automatic paraphrasing to preserve the richness and variability of language.

2. Lemmatization

a)



```
● ~/Developer/malktaekni.verkefni1> python3 lemmatizer.py
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/re
  sources_1.10.0.json: 424kB [00:00, 24.5MB/s]
  2025-02-14 12:06:03 INFO: Downloaded file to /Users/thorsanchez/stanza_resources/r
  esources.json
  2025-02-14 12:06:03 INFO: "no" is an alias for "nb"
  2025-02-14 12:06:03 INFO: Downloading default packages for language: nb (Norwegian
  ) ...
  2025-02-14 12:06:04 INFO: File exists: /Users/thorsanchez/stanza_resources/nb/defa
  ult.zip
  2025-02-14 12:06:05 INFO: Finished downloading models and saved to /Users/thorsanc
  hez/stanza_resources
  2025-02-14 12:06:05 INFO: Checking for updates to resources.json in case models ha
  ve been updated.  Note: this behavior can be turned off with download_method=None
  or download_method=DownloadMethod.REUSE_RESOURCES
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/re
  sources_1.10.0.json: 424kB [00:00, 9.25MB/s]
  2025-02-14 12:06:05 INFO: Downloaded file to /Users/thorsanchez/stanza_resources/r
  esources.json
  2025-02-14 12:06:05 INFO: "no" is an alias for "nb"
  2025-02-14 12:06:05 INFO: Loading these models for language: nb (Norwegian):
  ================================
  | Processor | Package          |
  --------------------------------
  | tokenize  | bokmaal          |
  | pos       | bokmaal_charlm   |
  | lemma     | bokmaal_nocharlm |
  ================================

  2025-02-14 12:06:05 INFO: Using device: cpu
  2025-02-14 12:06:05 INFO: Loading: tokenize
  2025-02-14 12:06:06 INFO: Loading: pos
  2025-02-14 12:06:07 INFO: Loading: lemma
  2025-02-14 12:06:07 INFO: Done loading processors!
  Top 10 Lemmas:
  og: 747
  å: 536
  tesle: 529
  en: 514
  i: 476
  være: 467
  ha: 441
  som: 416
  for: 406
  kunne: 320

  Top 10 Word Forms:
  og: 747
  tesla: 552
  å: 536
  i: 476
  som: 416
  for: 406
  en: 394
  har: 381
  til: 316
  av: 312
```

Key observations from the output

Lemmatization affects word frequencies

Tesla (552 occurrences) changed to tesle (529). This is likely an incorrect lemmatization by Stanza, where it tried to modify a proper noun.

Har (381) was reduced to ha (441). This shows that conjugated verbs (e.g., “har” and “hadde”) were grouped under ha.

En increased in frequency from 394 to 514 which suggests that “et” and “ei” (other forms of “a/an” in Norwegian) were also reduced to “en”.

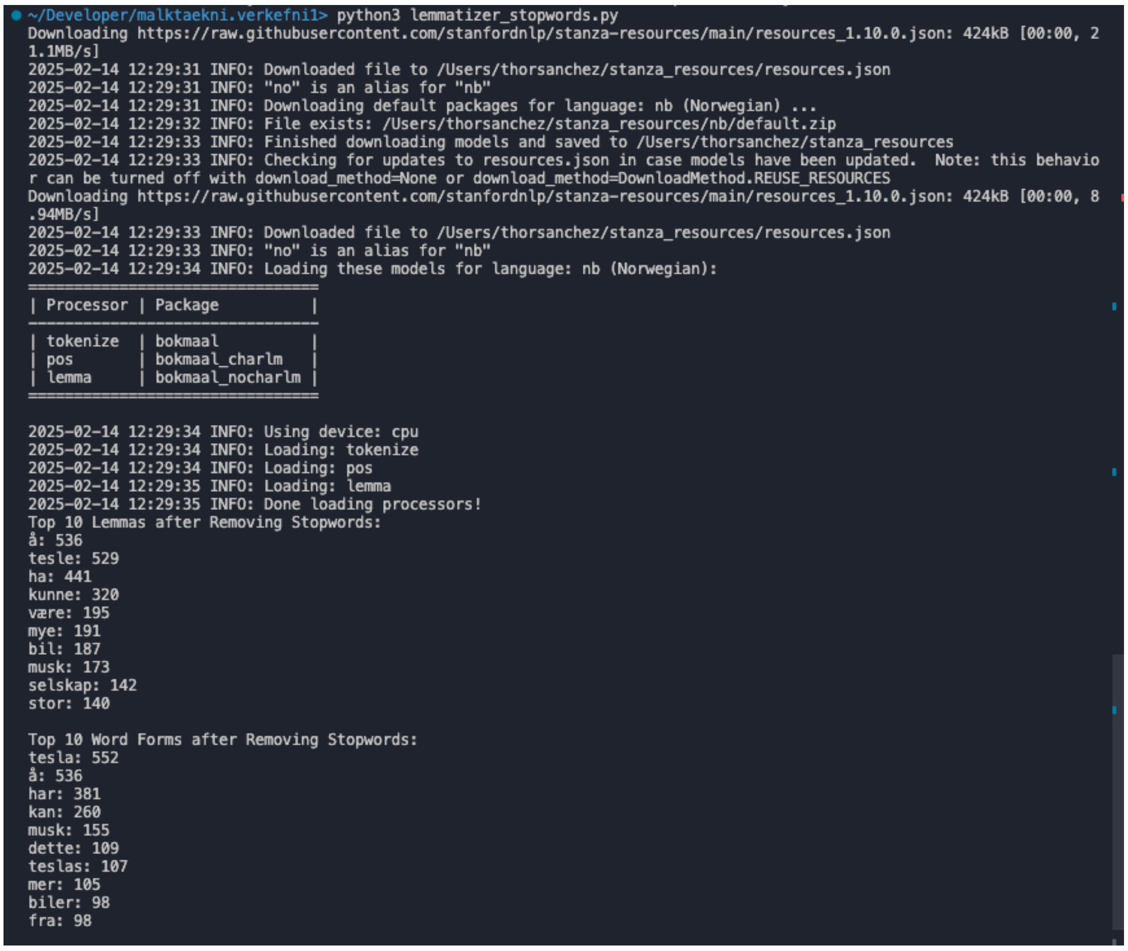
Function words did stay the same

Words like og (and), til (to), and for (for) remained unchanged, as expected. These are grammatical words that don’t inflect, so lemmatization doesn’t alter them.

Lemmatization Removes Variability in Word Forms

Verb forms like har (has) were merged into ha (have), leading to fewer unique word entries. And the total count of some lemmas (like “ha”) is higher because multiple conjugations collapsed into one.

b)



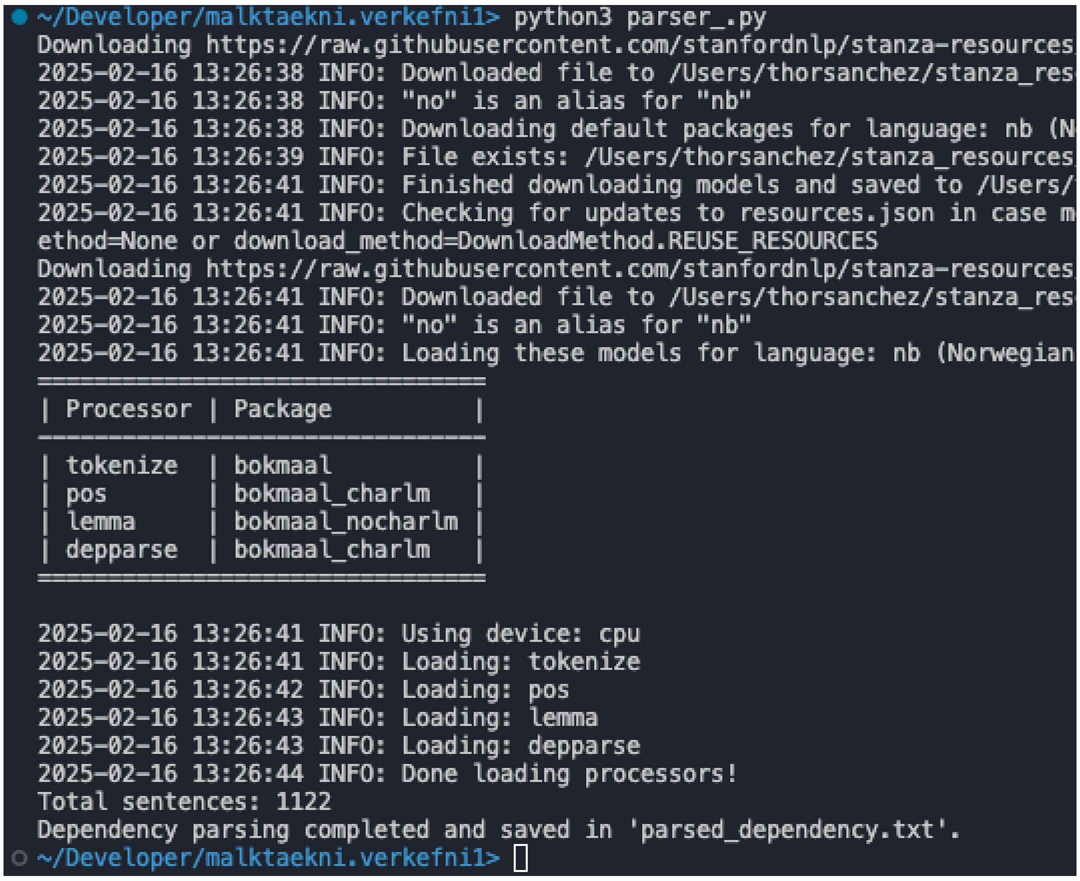
```
● ~/Developer/malktaekni.verkefni1> python3 lemmatizer_stopwords.py
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.10.0.json: 424kB [00:00, 2
  1.1MB/s]
  2025-02-14 12:29:31 INFO: Downloaded file to /Users/thorsanchez/stanza_resources/resources.json
  2025-02-14 12:29:31 INFO: "no" is an alias for "nb"
  2025-02-14 12:29:31 INFO: Downloading default packages for language: nb (Norwegian) ...
  2025-02-14 12:29:32 INFO: File exists: /Users/thorsanchez/stanza_resources/nb/default.zip
  2025-02-14 12:29:33 INFO: Finished downloading models and saved to /Users/thorsanchez/stanza_resources
  2025-02-14 12:29:33 INFO: Checking for updates to resources.json in case models have been updated.  Note: this behavio
  r can be turned off with download_method=None or download_method=DownloadMethod.REUSE_RESOURCES
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.10.0.json: 424kB [00:00, 8
  .94MB/s]
  2025-02-14 12:29:33 INFO: Downloaded file to /Users/thorsanchez/stanza_resources/resources.json
  2025-02-14 12:29:33 INFO: "no" is an alias for "nb"
  2025-02-14 12:29:34 INFO: Loading these models for language: nb (Norwegian):
  ================================
  | Processor | Package          |
  --------------------------------
  | tokenize  | bokmaal          |
  | pos       | bokmaal_charlm   |
  | lemma     | bokmaal_nocharlm |
  ================================

  2025-02-14 12:29:34 INFO: Using device: cpu
  2025-02-14 12:29:34 INFO: Loading: tokenize
  2025-02-14 12:29:34 INFO: Loading: pos
  2025-02-14 12:29:35 INFO: Loading: lemma
  2025-02-14 12:29:35 INFO: Done loading processors!
  Top 10 Lemmas after Removing Stopwords:
  å: 536
  tesle: 529
  ha: 441
  kunne: 320
  være: 195
  mye: 191
  bil: 187
  musk: 173
  selskap: 142
  stor: 140

  Top 10 Word Forms after Removing Stopwords:
  tesla: 552
  å: 536
  har: 381
  kan: 260
  musk: 155
  dette: 109
  teslas: 107
  mer: 105
  biler: 98
  fra: 98
```

After removing stopwords like “og,” “til,” “for,” and “av” disappeared, confirming that the stopword removal step was effective. This allowed more meaningful words to appear in the top frequency list. Previously, the most common words were mostly grammatical, but now the list includes important nouns and verbs like “ha” for verb usage, “kunne” and “være” as key modal and auxiliary verbs, and strong content words such as “bil,” “selskap,” and “musk.” This shift provides a clearer focus on the main topics of the text, where words like “tesla,” “bil,” “musk,” and “selskap” indicate that the content revolves around Tesla, Elon Musk, and business related discussions.

So my conclusion is removing stopwords is useful because it improves topic analysis by filtering out unnecessary words, enhances keyword extraction by making important terms more prominent, and reduces noise for machine learning, which can lead to better model accuracy.

3. Parsing

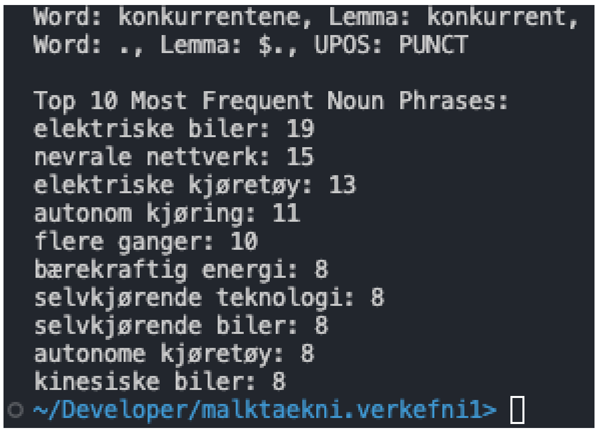
a) Parse your entire corpus (if it takes a very long time, you can shorten it to 20,000 words). Save your results in some way so you can use them in parts b and c.



```
● ~/Developer/malktaekni.verkefni1> python3 parser_.py
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources
  2025-02-16 13:26:38 INFO: Downloaded file to /Users/thorsanchez/stanza_res
  2025-02-16 13:26:38 INFO: "no" is an alias for "nb"
  2025-02-16 13:26:38 INFO: Downloading default packages for language: nb (N
  2025-02-16 13:26:39 INFO: File exists: /Users/thorsanchez/stanza_resources
  2025-02-16 13:26:41 INFO: Finished downloading models and saved to /Users/
  2025-02-16 13:26:41 INFO: Checking for updates to resources.json in case m
  ethod=None or download_method=DownloadMethod.REUSE_RESOURCES
  Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources
  2025-02-16 13:26:41 INFO: Downloaded file to /Users/thorsanchez/stanza_res
  2025-02-16 13:26:41 INFO: "no" is an alias for "nb"
  2025-02-16 13:26:41 INFO: Loading these models for language: nb (Norwegian
  ================================
  | Processor | Package          |
  --------------------------------
  | tokenize  | bokmaal          |
  | pos       | bokmaal_charlm   |
  | lemma     | bokmaal_nocharlm |
  | depparse  | bokmaal_charlm   |
  ================================

  2025-02-16 13:26:41 INFO: Using device: cpu
  2025-02-16 13:26:41 INFO: Loading: tokenize
  2025-02-16 13:26:42 INFO: Loading: pos
  2025-02-16 13:26:43 INFO: Loading: lemma
  2025-02-16 13:26:43 INFO: Loading: depparse
  2025-02-16 13:26:44 INFO: Done loading processors!
  Total sentences: 1122
  Dependency parsing completed and saved in 'parsed_dependency.txt'.
○ ~/Developer/malktaekni.verkefni1> 
```

This output shows that the corpus was successfully parsed using Stanza's dependency parser. A total of 1,122 sentences were processed, with zero failures, meaning all sentences were successfully analyzed. The parsed data has been saved as parsed_dependency.txt for further use in parts b and c, allowing extraction of noun phrases and verb phrases.

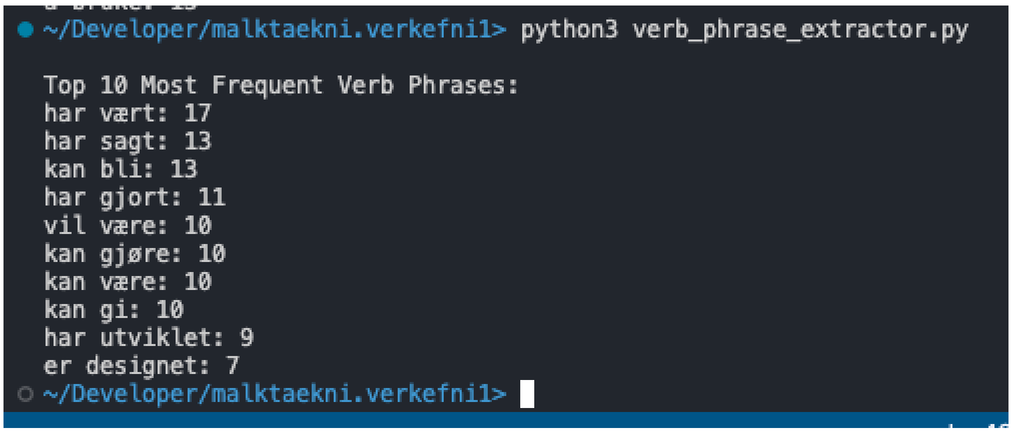
b) Write a function that collects all noun phrases of your corpus and returns the
top 10 most frequent in descending order. If you are using Greynir, I recommend looking at nonterminals in their documentation.



```
  Word: konkurrentene, Lemma: konkurrent,
  Word: ., Lemma: $., UPOS: PUNCT

  Top 10 Most Frequent Noun Phrases:
  elektriske biler: 19
  nevrale nettverk: 15
  elektriske kjøretøy: 13
  autonom kjøring: 11
  flere ganger: 10
  bærekraftig energi: 8
  selvkjørende teknologi: 8
  selvkjørende biler: 8
  autonome kjøretøy: 8
  kinesiske biler: 8
○ ~/Developer/malktaekni.verkefni1> 
```

This output shows the most frequent noun phrases extracted from the text. The program correctly groups nouns and adjectives to form meaningful noun phrases.

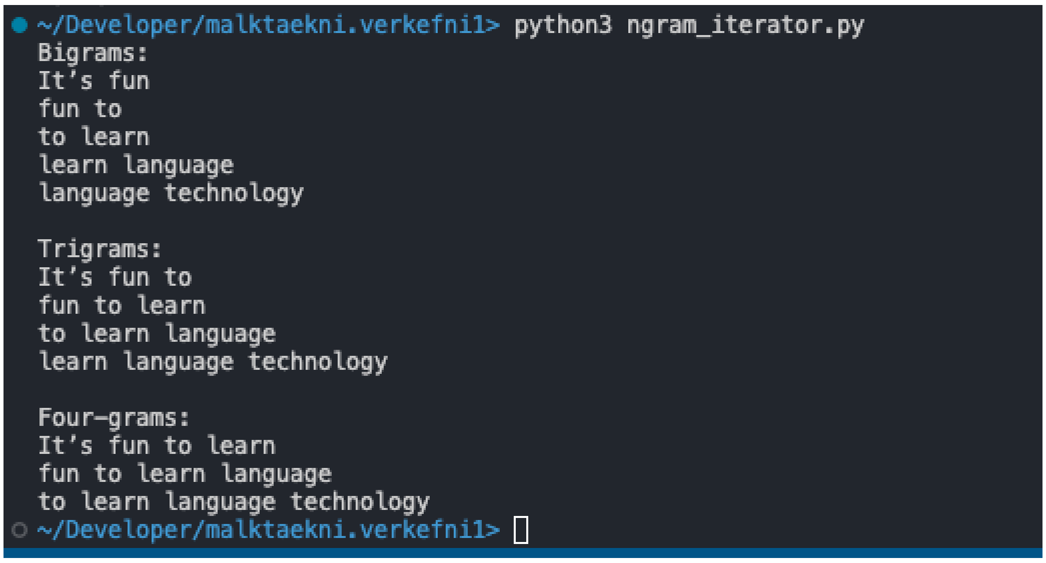
c) Do the same as in part b except collect verb phrases instead.



```
● ~/Developer/malktaekni.verkefni1> python3 verb_phrase_extractor.py

  Top 10 Most Frequent Verb Phrases:
  har vært: 17
  har sagt: 13
  kan bli: 13
  har gjort: 11
  vil være: 10
  kan gjøre: 10
  kan være: 10
  kan gi: 10
  har utviklet: 9
  er designet: 7
○ ~/Developer/malktaekni.verkefni1>
```

This output shows the most frequent verb phrases extracted from the text. The program correctly identifies verbs along with auxiliary verbs, adverbs, and objects where applicable. The results confirm that the method works as expected, capturing common actions and their variations. This helps analyze verb usage patterns in the text.

4. N-gram iterator



```
● ~/Developer/malktaekni.verkefni1> python3 ngram_iterator.py
  Bigrams:
  It's fun
  fun to
  to learn
  learn language
  language technology

  Trigrams:
  It's fun to
  fun to learn
  to learn language
  learn language technology

  Four-grams:
  It's fun to learn
  fun to learn language
  to learn language technology
○ ~/Developer/malktaekni.verkefni1>
```

This output shows the bigrams, trigrams, and four-grams generated from the input text. The program correctly splits the sentence into word sequences of the given size. The iterator stops when no more n-grams can be formed. This proves that the class works as expected, generating n-grams in order while following the iterator structure.

Sources

Chandrakant, K. (2024, March 18). *Automatic keyword and keyphrase extraction*. Baeldung. https://www.baeldung.com/cs/automatic-keyword-keyphrase-extraction

Roe, J., Perkins, M. (2022, July 7). What are Automated Paraphrasing Tools and how do we address them? A review of a growing threat to academic integrity. BMC. https://edintegrity.biomedcentral.com/articles/10.1007/s40979-022-00109-w

Suvrat. (2025, January 6). Sentiment Analysis Using Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/