Research Summary on Tokenization

1. **Definition**

   Tokenization is the process of breaking down raw text into smaller units called tokens. These tokens can be words, subwords, characters, or bytes, and they are essential for models or pipelines to process and manipulate text. In **Natural Language Processing (NLP)**, tokenization is the initial step that enables computers to comprehend human language before tasks such as translation, chatbots, or sentiment analysis. Inadequate tokenization can lead to significant confusion for the model.
2. **Types of Tokenization**
   - **Whitespace Tokenization:** Splits text based on whitespace characters. While simple, it has limitations as it does not account for punctuation or contractions (e.g., "I'm fine" becomes ["I'm", "fine"]).
   - **Rule-based/Regex Tokenization:** Employs predefined rules or regular expressions to define how text should be split, preventing splitting of numbers or keeping hyphenated words intact. It offers increased accuracy and flexibility but can become complex to maintain.
   - **Character-level Tokenization:** Treats every individual character or symbol as a distinct token. Beneficial for unusual spellings or local vocabulary, but results in very long token sequences, increasing computational burden.
   - **Subword Tokenization (BPE, WordPiece, SentencePiece):** Breaks down words into smaller, meaningful chunks (e.g., "running" becomes ["run", "##ning"]). Highly effective and widely adopted by modern NLP models like BERT and GPT, balancing character-level and word-level tokenization.
   - **Byte-level Tokenization:** Operates on individual bytes, making it highly versatile and capable of processing text in any language. Its robustness and broad applicability are seen in prominent Hugging Face and various GPT models.
3. **Examples of Widely-Used Tokenizers**
   - **NLTK:** A long-established and reliable option for basic text segmentation.
   - **spaCy:** Noted for its high processing speed and robust grammatical and rule-based handling across numerous languages.
   - **Hugging Face Tokenizers:** Commonly employed in transformer models (e.g., BERT, GPT), capable of processing subwords and bytes.
   - **SentencePiece:** Operates without pre-segmented text, highly effective for multilingual contexts or African languages.

4. **Advantages & Disadvantages**

Type	Pros	Cons
Whitespace	Simple, quick	Confused by punctuation
Rule-based	Smart if rules are good	Hard to maintain
Character	No unknown words	Not suitable for big text
Sub-word	Handles new words	Struggles with diacritics and tone
Byte-level	Works for all languages	Hard to interpret

5. Tokenization Challenges with Low-Resource Languages

- **Low data availability:** Insufficient clean text data for training effective tokenizers.
- **Orthographic variations:** A single word may have multiple acceptable spellings, especially across dialects.
- **Code-switching:** Integration of multiple languages within a single sentence (e.g., in Nigeria).
- **Agglutination:** Combining multiple words into a single linguistic unit, challenging standard tokenization.
- **Diacritics:** Marks like those in Yoruba (“ò,” “ó”) convey different meanings; some tokenizers fail to differentiate them.

6. Summary of Research Papers Reviewed

- **“Effect of Tokenisation Strategies for Low-Resourced Southern African Languages” — J. Rajab et al., Workshop on African NLP (ICLR workshop), 2022**
  - **Addressed:** How different subword tokenizers (subword-nmt BPE vs. SentencePiece variants) and hyperparameters affect neural machine translation (NMT) for Southern African (Bantu) languages.
  - **Solved:** Trained NMT models with various tokenizers and vocabulary sizes on low-resource datasets (Northern Sotho, Setswana, Xitsonga,

isiZulu), comparing BLEU scores and analyzing token splits/OOV behavior.
        - **Key findings:** Tokenizer choice and vocabulary size significantly impact downstream NMT performance; SentencePiece/unigram often provides robust results when carefully tuned.
        - **Why useful:** Provides direct experimental evidence that tokenizer hyperparameters are critical for low-data African languages, beyond default settings.
    - **“Language Model Tokenizers Introduce Unfairness” — NeurIPS poster/paper (2023)**
        - **Addressed:** Tokenization itself can introduce disparities across languages, with some languages receiving shorter token encodings (cheaper to process) and better coverage than others, leading to unfairness in compute cost and model access.
        - **Solved:** Measured tokenization length/fertility differences across languages for modern tokenizers, revealing large disparities (many-fold differences). Argued for fairer multilingual tokenizers.
        - **Key findings:** Tokenizers are not neutral; tokenization length and coverage vary widely across languages—an important consideration when working with low-resource African languages that may be penalized.