

Homework 3: Text Processing Fundamentals

Points: 20 | **Due:** Sunday, February 15, 2026 @ 11pm Pacific

Author: Richard Young, Ph.D. | UNLV Lee Business School

Compute: CPU (free tier)

Learning Objectives

1. **Install and use** NLP libraries (spaCy, NLTK) in Google Colab
 2. **Understand** WHY we preprocess text (not just how)
 3. **Create** domain-specific stopwords for your data
 4. **Measure** the impact of text cleaning on analysis
 5. **Identify** cases where cleaning hurts your analysis
-

Why This Matters for Business

Market Research: Before analyzing thousands of customer reviews, companies like Procter & Gamble preprocess text to focus on meaningful words. Poor preprocessing = misleading insights = bad product decisions.

Sentiment Analysis: When United Airlines analyzes social media mentions, removing “not” from “not satisfied” completely inverts the meaning. Smart text processing preserves business-critical signals.

Search & Discovery: E-commerce platforms like Shopify tune their stopword lists per industry. A wine retailer keeps “dry” (meaningful); a weather app removes it (noise).

Grading

Component	Points	Effort	What We're Looking For
Environment Setup	3	*	NLP libraries installed and working
Standard Stopwords	4	*	Applied and analyzed impact
Domain Stopwords	5	**	Created 10+ with justification
Negation Analysis	5	**	Smart stopwords preserving meaning
Visualization	3	*	Word cloud comparison
Total	20		

Effort Key: * Straightforward | ** Requires thinking | *** Challenge

The Big Picture

Text preprocessing is like preparing ingredients before cooking. But just like cooking, **the same preparation isn't right for every dish.**

Standard stopword removal can destroy important meaning (like negations in sentiment analysis).

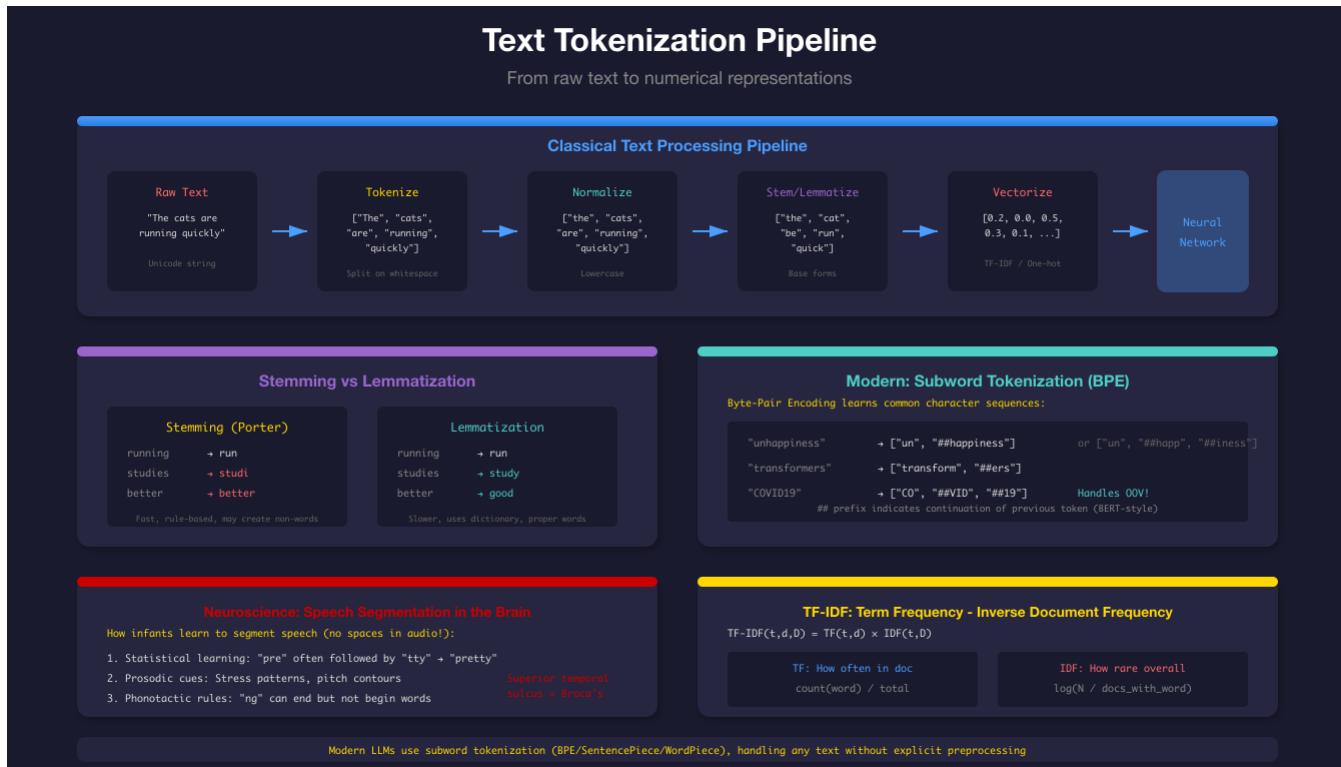


Figure 1: Text Tokenization Pipeline

Instructions

1. Open MIS769_HW3_Text_Processing.ipynb in Google Colab
2. Load your dataset (HuggingFace, Kaggle, or your own)
3. Apply standard stopword removal and analyze what was removed
4. Create your own domain-specific stopwords with justification
5. Analyze the negation problem and create a “smart” stopword list
6. Generate word cloud visualizations comparing approaches

What Your Output Should Look Like

Stopword Removal Impact:

STOPWORD REMOVAL IMPACT

Average reduction: 45.2%

Median reduction: 44.8%

Top Words After Cleaning:

- TOP 30 MOST COMMON WORDS (after standard cleaning)

1. movie	12,847
2. film	9,234
3. good	7,892
4. like	6,543
...	

Negation Problem Example:

Original: This product is not good at all
 Cleaned: product good ← MEANING REVERSED!

Common Mistakes (and How to Avoid Them)

Mistake	Symptom	Fix
Forgetting to lowercase	“Movie” and “movie” counted separately	Add .lower() before tokenizing
Removing all punctuation first	Can’t detect sentence boundaries	Keep punctuation until after sentence splitting
Using wrong stopword list	British vs American English issues	Check: ‘colour’ in stopwords
Not restarting runtime	ModuleNotFoundError after pip install	Runtime □ Restart runtime
Treating numbers as stopwords	Lose important data like prices, ratings	Keep numbers if relevant to your domain

If you see this error:

LookupError: Resource stopwords **not** found.

Run: `nltk.download('stopwords')`

Questions to Answer

- **Q1:** Which removed stopwords might carry meaning in your domain?
 - **Q2:** Why did you choose each domain-specific stopword?
 - **Q3:** When should you preserve vs. remove negations?
-

Going Deeper (Optional Challenges)

Challenge A: Lemmatization Comparison

Compare results using stemming (Porter) vs. lemmatization (spaCy). Which produces more interpretable results for business reporting?

Challenge B: Multi-language Stopwords

If your dataset contains non-English text, implement stopword removal for a second language. Compare the challenges.

Challenge C: Context-Aware Stopwords

Build a function that removes stopwords only when they appear in certain contexts (e.g., keep “not” before adjectives, remove it elsewhere).

Quick Reference

```
# Load libraries
import nltk
from nltk.corpus import stopwords
import spacy
nlp = spacy.load("en_core_web_sm")

# Get standard stopwords
STOPWORDS = set(stopwords.words('english'))

# Remove stopwords
def clean(text):
    words = text.lower().split()
    return ' '.join([w for w in words if w not in STOPWORDS])

# Preserve negations
NEGATIONS = {'not', 'no', 'never', 'neither', "n't", 'nor'}
SMART_STOPWORDS = STOPWORDS - NEGATIONS

# spaCy tokenization (better than split)
doc = nlp(text)
tokens = [token.text for token in doc]

# Word frequency
from collections import Counter
freq = Counter(all_words).most_common(20)
```

Useful spaCy Token Attributes: | Attribute | Description | Example | |-----|-----|-----|
 | token.text | Original text | “running” | | token.lemma_ | Base form | “run” | | token.pos_ | Part of speech
 | “VERB” | | token.is_stop? | Is stopword? | True/False |

Submission

Upload to Canvas: - Your completed .ipynb notebook with all cells executed

— *Richard Young, Ph.D.*