# POS Tagging Pipelines with NetworkX Visualization

## Introduction

This document presents a comprehensive analysis of Part-of-Speech (POS) tagging pipelines applied to Arabic text, accompanied by network graph visualizations using NetworkX. We explore four distinct approaches—rule-based (NLTK), statistical (CRF), spaCy's pretrained pipeline, and a Transformer-based pipeline (CAMeLBERT)—to evaluate their effectiveness and visualize their tagging structures.

## Data Description

* **Input Text**: A short Arabic paragraph describing the film *Interstellar*, containing proper nouns, verbs, and syntactic constructs typical of narrative descriptions.
* **Training Data**: For the CRF pipeline, the English Penn Treebank corpus (universal tagset) was used to train the model. The other pipelines rely on pretrained resources and do not require additional training data.

## Tools and Libraries

* Python 3
* **NLTK**: Tokenization and rule-based POS tagging
* **sklearn-crfsuite**: Conditional Random Field modeling
* **spaCy**: Pretrained English pipeline
* **Transformers (Hugging Face)**: CAMeLBERT Arabic POS tagger
* **NetworkX & Matplotlib**: Graph construction and visualization

## Baseline Experiment: Rule-Based Pipeline (NLTK)

**Goal:** Use NLTK's pos\_tag with the universal tagset to tag Arabic text and visualize word-to-tag relationships.

**Procedure:**

1. Tokenize text with word\_tokenize.
2. Tag tokens using pos\_tag(..., tagset='universal').
3. Build a DataFrame of (word, tag) pairs.
4. Construct a directed multigraph (MultiDiGraph) with words as source nodes and tags as target nodes.
5. Color-code nodes: skyblue for words, lightgreen for tags.
6. Plot with a spring layout.

**Results & Conclusion:**

* Correctly tags named entities but fails to capture Arabic morphology and context. Useful as a simple baseline but limited in accuracy for non-English text.

## Experiment 2: Statistical CRF Pipeline (sklearn-crfsuite)

**Goal:** Leverage a CRF trained on the Penn Treebank to improve contextual POS tagging.

**Procedure:**

1. Define word2features to extract morphological and contextual features.
2. Convert Treebank sentences into feature sequences (X\_train) and label sequences (y\_train).
3. Train a CRF model with L1/L2 penalties (c1=0.1, c2=0.1).
4. Apply tokenization and feature extraction to Arabic text.
5. Predict POS tags and build the network graph.

**Results & Conclusion:**

* Achieves better contextual tagging than rule-based. However, performance is constrained by training on English data, leading to mismatches in Arabic syntax and vocabulary.

## Experiment 3: spaCy Pretrained Pipeline

**Goal:** Evaluate spaCy's en\_core\_web\_sm pipeline on Arabic input.

**Procedure:**

1. Load en\_core\_web\_sm and process Arabic text.
2. Extract (token.text, token.pos\_) pairs.
3. Visualize with NetworkX as before.

**Results & Conclusion:**

* Most tags are inaccurate or omitted due to language mismatch. Highlights the necessity of language-specific models for reliable performance.

## Experiment 4: Transformer Pipeline (Hugging Face)

**Goal:** Utilize CAMeL-Lab’s camelbert-ca-pos-egy model fine-tuned for Arabic POS tagging.

**Procedure:**

1. Load the tokenizer and model via AutoTokenizer and AutoModelForTokenClassification.
2. Create a token-classification pipeline with aggregation\_strategy='simple'.
3. Tag the text and aggregate subword pieces into full words.
4. Build and visualize the directed graph.

**Results & Conclusion:**

* Demonstrates superior accuracy and contextual understanding for Arabic text. Correctly captures morphological and syntactic nuances.

## Overall Conclusion

* **Rule-Based (NLTK):** Simple and fast, but poor accuracy on Arabic.
* **CRF (sklearn-crfsuite):** Improved context-awareness, limited by language mismatch.
* **spaCy:** Inadequate without Arabic-specific models.
* **Transformers (CAMeLBERT):** Best overall performance for Arabic POS tagging.

Network graph visualizations using NetworkX offer an intuitive view of word-tag relationships and highlight differences among tagging approaches.

## External Resources

1. NLTK Documentation and Penn Treebank corpus
2. sklearn-crfsuite GitHub
3. spaCy Documentation
4. Hugging Face Transformers and CAMeLBERT model

## Reflection Questions

1. **Biggest Challenge:** Applying models trained on English to Arabic text and engineering features for CRF.
2. **Key Learnings:** Construction of diverse POS pipelines, feature engineering for CRF, and the power of transformer-based models for cross-lingual tasks.