L04_Journal_ImanHaamid_ITAI_2373

Date: June 26, 2025
So, Lab 04 on Text Representation was a real eye-opener! It's all about how we teach computers to "get" human language, which is super tricky because our language is a messy, unstructured thing. This lab hammered home that how well an NLP model works totally depends on how we represent text data.

Before this, I kind of knew computers didn't "read" like we do, but the nitty-gritty of turning words into numbers was a mystery. This week, we dug into a few key methods, each with its own pros and cons, all aiming to bridge that gap between human talk and machine code.

First up was **Bag-of-Words (BoW)**, which is super simple: it just counts how often words pop up in a document, ignoring grammar and even word order. You get a vocabulary from all the text, and each document becomes a list of numbers where each number is how often a word from the vocabulary shows up. It's okay for basic stuff like classifying text, but the big downside hit me right away: it completely ignores **context**. "Man bites dog" and "Dog bites man" would look identical to BoW, which is a problem if you want any real understanding. Plus, those lists of numbers can get really long and empty, especially with huge vocabularies.

Building on BoW, **TF-IDF (Term Frequency-Inverse Document Frequency)** was a cooler approach. It's smarter about weighting words, not just by how often they're in a document (TF), but also by how *important* they are across all the documents (IDF). The idea is that common words (like "the" or "is") aren't as useful as words unique to a specific document. TF-IDF helps highlight those unique, important terms, making it better for finding information or keywords. It's definitely an upgrade from just counting words because it tries to capture a word's relevance, but it still misses out on context, just like BoW.

The coolest part of this lab for me was diving into **Word Embeddings**. This is where text representation gets seriously interesting and powerful. Unlike the long, empty lists from BoW and TF-IDF, word embeddings (like those from **Word2Vec** or **GloVe**) create short, packed lists of numbers. The magic is that these numbers are designed to capture **semantic relationships** between words. Words with similar meanings or contexts are closer together in this numerical space. The classic "King - Man + Woman = Queen" example really shows how these embeddings learn not just what words are there, but their hidden connections and meanings based on how they're used in tons of text. This ability to capture context and meaning makes embeddings super valuable for advanced NLP tasks like figuring out sentiment, translating languages, and answering questions. It feels like a leap from just counting to actually "understanding" language on a deeper level.

Looking back at all the methods, it's clear there's no one-size-fits-all solution. For simpler tasks with less data, BoW or TF-IDF might be fine because they're easy. But for more complex, meaning-driven tasks, word embeddings are a must-have. You need a ton of data to train good word embeddings, but the payoff in how well your model performs is often huge.

This lab totally changed how I see AI working with human language. It's not just about throwing text at an algorithm; it's about carefully turning that text into numbers that keep as much of its meaning and context as possible. The idea of turning abstract language into a tangible, movable math space is just brilliant. Understanding these text representation techniques is fundamental for anyone wanting to build or even just grasp what modern AI systems can and can't do with human language. I'm especially stoked to see how these basic ideas play out in more advanced NLP stuff in future labs!