Part-of-Speech Tagging using Contextual Word Embeddings

Juhani Dickinson
jdickin9@uwo.ca

Paul Moore
email address or ORCID

***Abstract*—ABSTRACT**

***Index Terms*—pos tagging, part of speech, nlp, word embedding, contextual word embedding**

I. Introduction

A. *Parts of Speech*

In a sentence, each word has a syntactic role to play. In the sentence “The brown dog runs quickly towards food.”, “dog” denotes a thing, “brown” modifies or describes “dog”, “runs” denotes an action, and so on. These syntactic roles are commonly referred to as “parts of speech”, and are mapped to words using part-of-speech tags. However, this is not a one-to-one mapping. Many words have multiple meanings, and in many cases, these different meanings play different syntactic roles, mapping them to different parts of speech. For example, the word “orange”, spelled and pronounced identically, can be either a noun or an adjective, as seen in sentences (1) and (2).

“I often eat an orange after working out.” (Noun) (1)

“The orange car has arrived.” (Adjective) (2)

Figuring out what part of speech a word has is not an unsolvable task, however. Where a word is in the sentence and what words surround it play a major role in disambiguating between syntactic roles. In (1), the word after “orange” is “after”, which is an adverb. “Orange” is either an adjective or a noun, and since the following word is an adverb, it cannot be an adjective. In (2), the words surrounding “orange” are “the” and “car”, which are a determiner and a noun respectively. In this position, “orange” describes “car”, making it an adjective.

B. *Part-of-Speech (POS) Tagging*

The task of POS tagging (often simply called tagging) is as follows: given a sentence, label each word in the sentence with the POS tag that describes its syntactic role, seen below for sentence (3).

(3)	Can	you	spot	the	large	spot	on	your	nose	?
	AUX	D	VB	D	ADJ	N	PP	D	N	.

What makes the task of tagging challenging is the contextual requirement: without it, both instances of “spot” in sentence (3) would be tagged as either VB or N. Because of this, non-naive tagger implementations leverage some level of context to perform their task. Simple implementations like n-gram taggers assign tags to words based on both the word itself and the tags of the preceding n words.

C. *Word Embeddings*

A common issue with simple tagging implementations is resistance to parallelisation. Any operation below the level of sentence must be linear, as the tag of the current word depends on the tag(s) of the previous words. To get around the linear-time requirement, some taggers use word embeddings, which represent words as vectors. This embedding takes into account the surrounding words, so that the meaning and context of each word is stored in the embedding. Using embeddings, tagging can be parallelised at the level of individual words. ¡–HOW DO THEY WORK?–¿

1) Static Word Embeddings: While word embeddings take context into account by design, static word embeddings take the naive approach to context. These mapping functions collect all contexts for a given word and uses them to generate a single embedding per word. Often, this approach fails to take into account that words can have different meanings (“river bank” vs. “bank account”) or different syntactic roles. This means that a singular word embedding must hold information on every possible context for any given word, which causes problems when the different meanings of a word are not syntactically or semantically close to one another.

2) Contextual Word Embeddings: Contextual word embeddings solve the syntactic and semantic problems that static word embeddings have by moving from word embeddings to word token embeddings. Now, words with multiple meanings or syntactic roles are split into multiple word tokens, one for each use. Contextual embeddings map these tokens to vectors, preserving specific meaning. With contextual embeddings, a word like “spot” is first split into two tokens, such as “spot–noun” and “spot–verb”, and these tokens are embedded separately. While contextual embedding systems are more computationally demanding to train and use, their context-aware nature often leads to increased performance in context-sensitive tasks such as POS tagging.

II. Related Work

pass

III. Motivation

Part-of-speech tags are useful and sometimes key pieces of information in many natural language processing (NLP) tasks such as text-to-speech (TTS), word-sense disambiguation, marking word order, and named entity recognition (¡ref¿). In TTS applications, the pronunciation of a word can change depending on what its syntactic role is or sometimes even what tense it is. The word “resume” is pronounced differently based on whether it is meant as “continue doing” or as “a document detailing professional experience”, which can be easily disambiguated based on whether it acts as a verb or a noun. This disambiguation-by-tag is also used in word-sense disambiguation, like in sentence (3). In a search-engine setting, understanding whether “spot” refers to an activity or a visual marker allows for more accurate search results.

While high-resource languages like English no longer use POS tagging in some popular NLP applications like AI assistants by making use of the massive collections of curated data now available, most languages do not have the luxury of large, high-quality datasets. In these low-resource languages, POS tags are still critical for NLP tasks.

For a task like machine translation, (¡ref¿) found that using linguistic features derived from POS tags improved the translation performance not only between the low-resource language pair of Thai and Myanmar but also between Thai and English and Myanmar and English. ¡–MORE–¿

¡–INTRO SENTENCE–¿. In low-resource languages where training data is scarce, one established method of finding training data is to use cross-linguistic transfer to automatically generate POS training data. This is done by lining up texts that appear in both a high-resource language and the target low-resource language and running POS taggers and parsers for the high-resource language (¡ref¿). The results from the taggers and parsers are then projected onto the low-resource text. ¡–TALK ABOUT RESULTS–¿

As well as being useful, the efficacy of cross-linguistic transfer is predictable. (¡ref¿) found that language families and morphological structures have a major impact on the performance of cross-linguistic performance. ¡–TALK ABOUT RESULTS–¿.

pass

Finding the a good POS tagger for English, a high-resource language, has clear benefits for linguistically-related low-resource languages due to the success of cross-linguistic transfer. One example is Scots, a language in the Anglic family currently considered “vulnerable” by the UNESCO Atlas of the World’s Languages in Danger (¡ref¿). Research suggests that NLP can support and help revitalise endangered languages (¡ref¿) through machine translation, TTS, and integration into learning materials. ¡–MORE–¿

To this end, we will test multiple prominent word embedding models on the same English dataset in order to determine the best model. Knowing the best model ¡–FINISH THIS–¿ For this study, word embeddings have been chosen over the recently-popular GPT-style architectures for their size, ability to be fine-tuned for very high accuracy, and customisation, while still being able to provide accurate tags (¡ref¿). Lightweight neural models that use embeddings such as BiLSTM-CRF often take less than 50MB of space, and BERT-based models often take between 300-600MB of space, making it feasible for them to be deployed and run directly on-location. On the other hand, GPT models are typically very large and very computationally demanding, meaning they often cannot be run on location and must be cloud-based. In addition to the space-efficiency being useful for in-the-field or near-the-field research, this also makes them accessible to language communities that have less powerful hardware. While embedding-based models need to be trained or fine-tuned to produce good results, after fine-tuning they perform better than GPT models, with BERT-based models achieving above 97% tagging accuracy, and lighter models providing competitive tagging accuracy (¡ref¿). While GPT models do not need to be trained, they have a lower accuracy on tagging, around 90-93%. The ability to train and fine-tune embedding-based models means that they can be directly customised to work with new data, domains, languages, and styles, which is very useful in academic settings or in settings that need specific solutions. While prompt engineering can get GPT models to work better for specific tasks, the lack of training ability and limited fine-tuning capability means it lacks the fine-grained control often needed for task-specific applications (¡ref¿).

IV. Methods

pass

V. Experimental Results

pass

VI. Conclusion

pass

References

[1] Pass