Things Smaller than Words

David R. Mortensen

May 16, 2023

Introduction

When we want to process language, we need to represent it as numbers and, since we assume that it is linear, we usually want to represent it as a sequence of numbers. But how should a passage of text be represented? In particular, how should it be divided into units that can be processed computationally?

To many people, the most obvious unit into which text can be divided is the word. In this scheme, the utterance

(1) Words aren't simply atomic entities!

would have five basic units. As we will learn, defining WORD is not trivial. Even in our example, there is a complication: *aren't* can be seen as either a single word or as the combination of two words, *are* + *not*. In this instance, *n't* is a special kind of word called a clitic which latches on to a neighboring world. If we segment off *n't*, then we get a sequence of six units:

(2) Words are n't simply atomic entities!

This has the benifit of given all of the texts that have *n't* similar representations, even when *n't* is part of *didn't*, *isn't*, *wasn't*, *weren't*, as so on. It effectually reduces the sparcity of the text data, which is good, but it doesn't scratch the surface of the problem. The frequencies of words roughly follow a power law distribution A few words are very frequent but most words are very infrequent. This means that we cannot learn a good representation of most words, if we are building models using words as tokens. It would be nice, both from a computational and a theoretical perspective, if we could analyze text into units that did not display this kind of distribution.

This is known as Zipf's Law, stated simply $f(r) \cong \frac{0.1}{r}$ where r is the rank order of a word. In other words, the frequency of the word with rank order r is approximately 0.1 over r.

The obvious answer is to tokenize by letter or character.

(3) `W o r d s _ a r e n ' t _ s i m p l y _ a t o m i c _ e n t i t i e s !`

While this may not be as appealing for languages written with LOGOGRAPHIC writing systems like Chinese (where the number of possible characters is very large and were the distribution of characters is roughly Zipfian), it addresses some aspects of tokenization for languages written with ALPHABETIC systems well. English, for example, has only 26 letters plus a handful of punctuation and whitespace characters. Every word (with exception like emoji) will be made up of strings of these characters.

One might even tokenize by phonemes (individual units of sound) in which the sequence would look like:

(4) w r̩ d z ɑ ɹ n̩ t s ɪ m p l i ʌ t ɑ m ɪ k ɛ n t ɪ t i z

Duality of Patterning

But the character-level and phoneme-level tokenizations run up against a problem. As Charles Hockett noted in his classic textbook, *A course in modern linguitics*, language and other sign systems display "duality of patterning"[1]. The observation is that the smallest units in language that can contribute to making distinctions in meaning are (by themselves) meaningless. For alphabetic writing, these are letters; for speech, these are phonemes. A similar observation was made by Martinet[2]. The upshot, from a computational perspective, is that embeddings of ⟨t⟩[3] are never going to be as meaningful as embeddings of tokens like ⟨n't⟩. ⟨n't⟩ only occurs in sentences that include 'NEGATIVE' as one their components but ⟨t⟩ can occur in almost any sentence.

[1] Charles F. Hockett. *A course in modern linguitics*. Macmillan, New York, 1958

[2] André Martinet. *Eléments de linguistique générale*. Colin, Paris, 1960

[3] Letters enclosed in angle brackets are GRAPHEMES—units of written language.

Duality of patterning states that there is a basic level of units that are distinctive but meaningless and that there is a more complex level of units that are both meaningful and possess internal structure (they're made out of the units from the basic level). If we could tokenize text into units on the second level, each token would be meaningful and could be assigned a meaningful embedding. In principle, in fact, we should be able to tokenize a sentence so that each token in a sentence is maximally informative regarding what the sentence means.

Meaningful units and how they combine will be discussed more in Lecture 2.

Tokenization Schemes

In NLP, various means have been developed to tokenize text below the level of the word, all of which will be discussed in greater detail subsequently. The simplest of these is byte pair encoding, a version of Huffman Coding introduced to NLP by Rico Sennrich[4]. Like the subsequent tokenization algorithms mentioned here, it is sensitive to the hyperparameter VOCABULARY SIZE. Our sentence, when tokenized with the BPE model employed with a particular version of the RoBERTa model is as follows:

Tokenization algorithms will be discussed in detail in Lecture 6.

[4] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. DOI: 10.18653/v1/P16-1162. URL `https://aclanthology.org/P16-1162`

(5) `Words Ġar en 't Ġsimp'ly Ġato mi c Ġen titi es !`

Where `Ġ` indicates the beginning of a non-inital "word." Wordpiece is a closely related tokenization scheme.[5] The Wordpiece model used in BERT yields results like:

[5] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016

(6) `words aren ' t simply atomic entities !`

The fact that there are fewer tokens results, in part, from the fact that the vocabulary size is larger relative to the set of characters.

Sentencepiece is a rather different tokenization scheme that does not assume that words are separated by whitespace.[6] The same sentence tokenized by the sentencepiece tokenizer used with certain versions of RoBERTa looks like this:

[6] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. DOI: 10.18653/v1/D18-2012. URL `https://aclanthology.org/D18-2012`

(7) `▁ Words ▁ aren't ▁ simply ▁ atomic ▁ entities!`

That is, except for the treatment of whitespace, it looks just like Example (1). Sentencepiece is based in part on a Unigram model of tokenization. When applied to our sentence, Unigram yields results like:

(8) `▁Word s ▁are n ' t ▁simply ▁atomic ▁ entitie s !`

The dramatic difference seen here between Sentencepiece and Unigram is due, largely, to differences in the vocabulary size of the models rather than the algorithms themselves. Nevertheless, it provides a convenient opportunity for comparison.

Morphemes

In the Unigram output, you probably noticed something you didn't notice in output of the earlier tokenizers: the ⟨s⟩ at the ends of ⟨Words⟩ and ⟨entities⟩ is segmented off as a separate token. Also, ⟨are⟩ is treated as a token independent of ⟨n⟩, ⟨'⟩, and ⟨t⟩ (which follow it). This is because Unigram captures an insight from linguistics that is not captured by the BPE, wordpiece, or sentencepiece tokenizers: ⟨Word⟩, ⟨are⟩, and ⟨entitie⟩ are all MORPHEMES (the units on Hockett's second tier). The plural ⟨s⟩ suffixes are also morphemes.

When tokenized into morphemes, our sentence would be as in (9):

(9) `▁Word s ▁are n't ▁simp ly ▁atom ic ▁entitie s`

Some observations:

(10) *A morpheme is a minimal meaningful unit.*
  a. By definition, a morpheme is something that can have a meaningful embedding
  b. All words are composed of morphemes
  c. Rare words are usually composed of more frequent morphemes

(11) Hypothesis:
  a. An optimal tokenization is one in which each token consists of exactly 1 morpheme.
  b. The next most optimal tokenizations are those in which each token consists of exactly *n* morphemes.

Allomorphy

Note something interesting about *simply* and *entities* in (9) above:

(12) a. The root *simple* is spelled here as ⟨simp⟩ (followed by the suffix ⟨ly⟩)
  b. The root *entity* is spelled here as ⟨entitie⟩ (followed by the suffix ⟨s⟩)

These are examples of the phenomenon called ALLOMORPHY[7].

Hypothesis:

[7] Allomorphy is when the same morpheme has different spellings or pronunciations based on context. It will be discussed more in Lecture 7.

(13) In an optimal tokenization, tokens are normalized such that all instances of a particular morpheme have the same representation (and the same embedding)

Looking Forward

In this lecture, we have already briefly introduced several of the themes of this Module:

- Meaningful units (signs) below the level of the word (morphemes)
- Morphological segmentation
- Computational approaches to segmentation and their relationship to tokenization
- Allomorphy

We will also talk about a number of related topics:

- Inflection, derivation, and compounding (morphological functions)
- Morphological operations on form besides affixation

AFFIXATION is a general term for prefixation and suffixation.

The First Project

The goal of the first project is to develop a tokenizer—a morphological segmenter—that segments text from an arbitrary language more similarly to gold morpheme segmentations by a linguist than two baselines: a rule-based (FST) baseline and a Unigram tokenization model.

The data

You will be provided with data from two languages:

- Rarámuri (Tahumara) [tar], an indigenous language of Northern Mexico that is part of the Uto-Aztecan language family.
- Shipibo-Konibo [shp], a Panoan language of the Peruvian Amazon.

For each language, for each set (train, dev, and test), you will be provided with a source file. You will be provided the target file for the dev set[8] The source file will consist of unsegmented words, one per line. The target fill will consist of the same words in the same order, but with the morphemes delimited by spaces, as shown in (14).

[8] The train and test target file will be held out.

(14) The first entries from the Shipibo-Conibo source and target data

a. `src: ointiyamaai`

b. `tgt: ointi yama ai`

The task

The goal is the reproduce the reference target given the source and a model trained on the (source side of the) training set. This can be seen an a sequence labeling task in which your model must decide whether a character is the beginning of a new span (morpheme) or the continuation of an existing span. There are other ways of viewing the task, but evaluation (in terms of precision/recall/F1) will be based on this paradigm. Labeling a character as "beginning" will be treated as a positive and not labeling it as such will be treated as a negative.

Baselines

As will often be the case in this course, there will be two baselines: one data-driven and one rule-based.

FST (rule-based). For each language, there will be a hand-crafted finite-state transducer based segmentation model. This model will be based on linguist analysis of the data and secondary literature on the morphology (word structure) of the language. You will be provided with access to the FST outputs for the dev set.

Unigram (data-driven). For each language, a Unigram model will be trained on the training set. If other unlabeled text data can be found, it will be used as well (and shared with you). You will be provided with the Unigram tokenizer output for the dev set.

Suggestions

- Developed improved information-theoretic tokenization algorithm in the same family as BPE or Unigram tokenization
- Develop better means of selecting an optimal vocabulary for BPE-like algorithms[9]
- Improve on unsupervised morphological segmentation algorithms like Morfessor and Morfessor FlatCat[10].
- Build a better rule-based (probably FST-based) morphological segmenter, perhaps using LEXC and Foma[11].

Extensions

- Does your segmentation/tokenization method provide better performance on some downstream tasks than widely used tokenization schemes like BPE, Unigram, and sentencepiece?
- More generally, is there a correlation between how well a tokenization scheme performs on the morphological segmentation task and how well it performs on downstream NLP tasks?

[9] Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online, August 2021. Association for Computational Linguistics. DOI: 10.18653/v1/2021.acl-long.571. URL `https://aclanthology.org/2021.acl-long.571`

[10] Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1177–1185, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL `https://aclanthology.org/C14-1111`; and Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3944–3953, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://aclanthology.org/2020.lrec-1.486`

[11] Mans Hulden. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, pages 29–32, Athens, Greece, April 2009. Association for Computational Linguistics. URL `https://aclanthology.org/E09-2008`

- How does the runtime efficiency of your method compare with the other methods?

References

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1177–1185, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL `https://aclanthology.org/C14-1111`.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3944–3953, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://aclanthology.org/2020.lrec-1.486`.

Charles F. Hockett. *A course in modern linguistics*. Macmillan, New York, 1958.

Mans Hulden. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, pages 29–32, Athens, Greece, April 2009. Association for Computational Linguistics. URL `https://aclanthology.org/E09-2008`.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. DOI: 10.18653/v1/D18-2012. URL `https://aclanthology.org/D18-2012`.

André Martinet. *Eléments de linguistique générale*. Colin, Paris, 1960.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. DOI: 10.18653/v1/P16-1162. URL `https://aclanthology.org/P16-1162`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online, August 2021. Association for Computational Linguistics. DOI: 10.18653/v1/2021.acl-long.571. URL `https://aclanthology.org/2021.acl-long.571`.