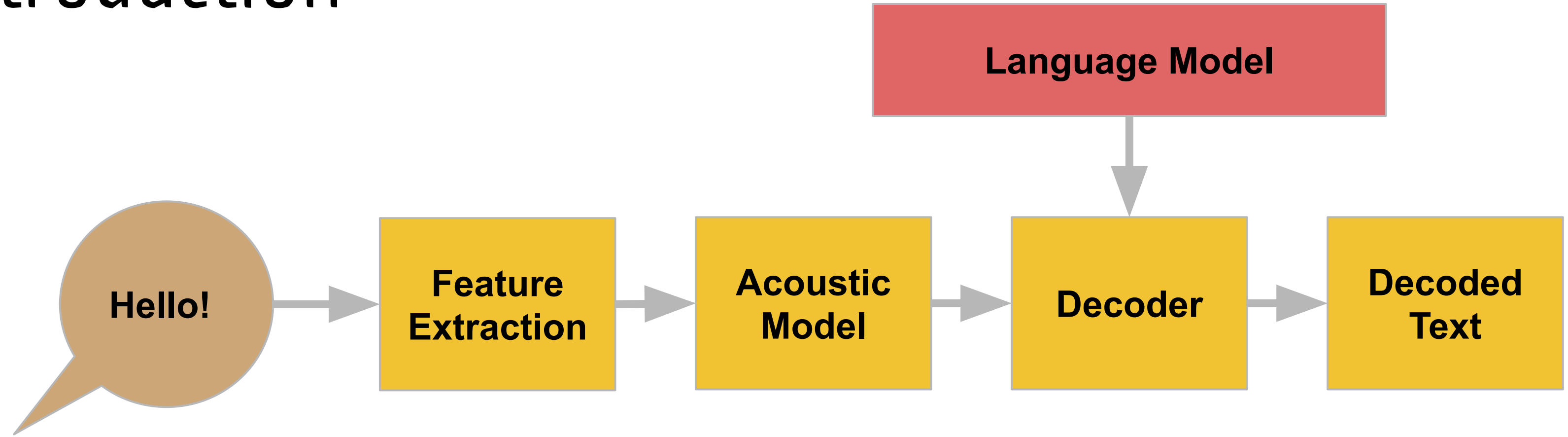
Subword Language model

Speech Recognition
Phung Duc Thao
Markus Ylisiurunen
Duong Hai Ly

Outline

- Introduction
- Language modeling (LM)
- Subword Tokenizations
- Subword Algorithms
- Experiments & Results
- Improvements
- Conclusion
- References

Introduction



- Feature extraction: MFCCs,
- Acoustic model **P(Y|ω)**: phoneme mapping
- Language model **P(ω)**: probability of input sequences
- Decoder: $\widehat{\omega} = \underbrace{\text{argmax}}_{\omega} \{P(Y|\omega)P(\omega)\}$

Language modeling

Subword-based language models:

- Neural network language models (BERT, GPT-2)
- N-gram language models: **$P(W)=P(\omega 1)P(\omega 2)P(\omega 3)\ldots P(\omega N)$**

  "Tonight I am making ___" ("dinner" or "breakfast")

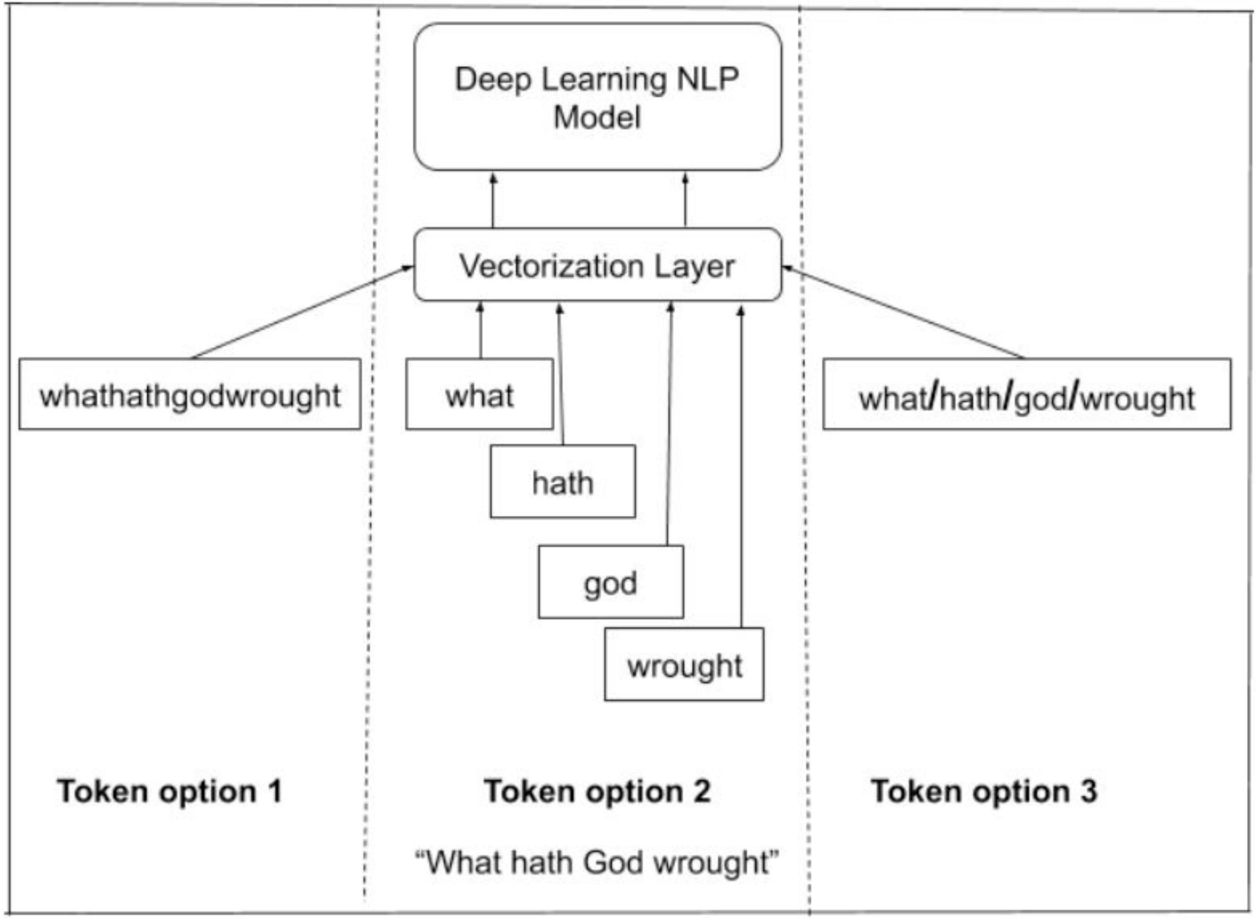
  $P(\text{dinner}|\text{Tonight I am making}) > P(\text{breakfast}|\text{Tonight I am making})$
- Examples

How to Wreck a Nice **? 1-gram: $P(\omega 6)$?**

How to Wreck a **Nice ? 2-gram: $P(\omega 6|\omega 5)$?**

How to Wreck **a Nice ? 3-gram: $P(\omega 6|\omega 5,\omega 4)$?**

Training Model



Training Model

The models have no knowledge of the language or input sequences

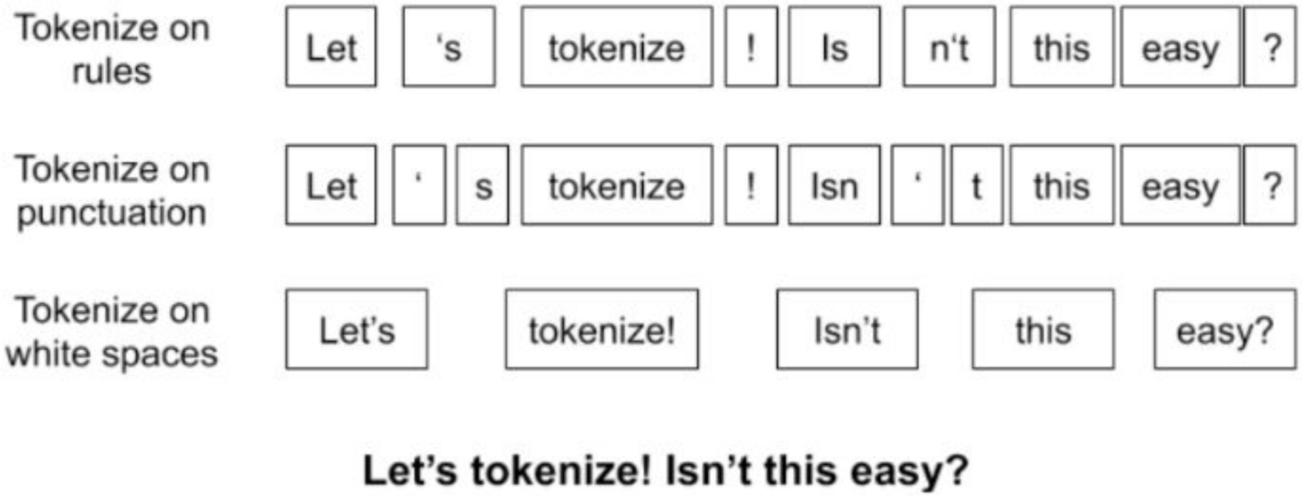
To train the models by following steps:

+ Split the input text into smaller chunks
+ Represent these inputs as vectors

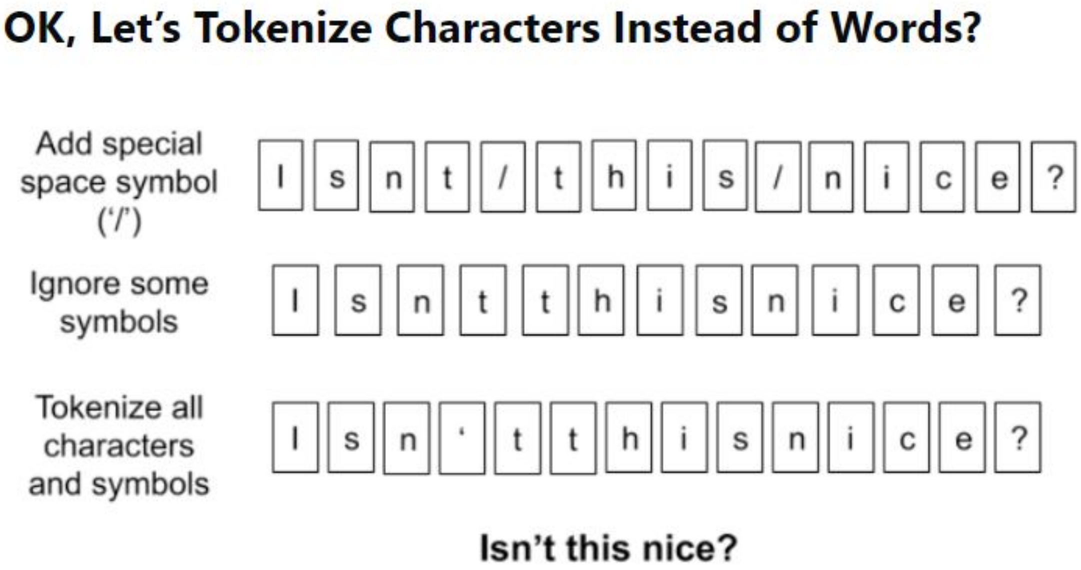
Drawbacks of word-based tokenization:

+ Rare or unseen words
+ Special characters
+ Big vocabulary
+ Morphological language

Word-level Tokenization



Character-level Tokenization



Character-level Tokenization

What if the characters is tokenized instead of words?

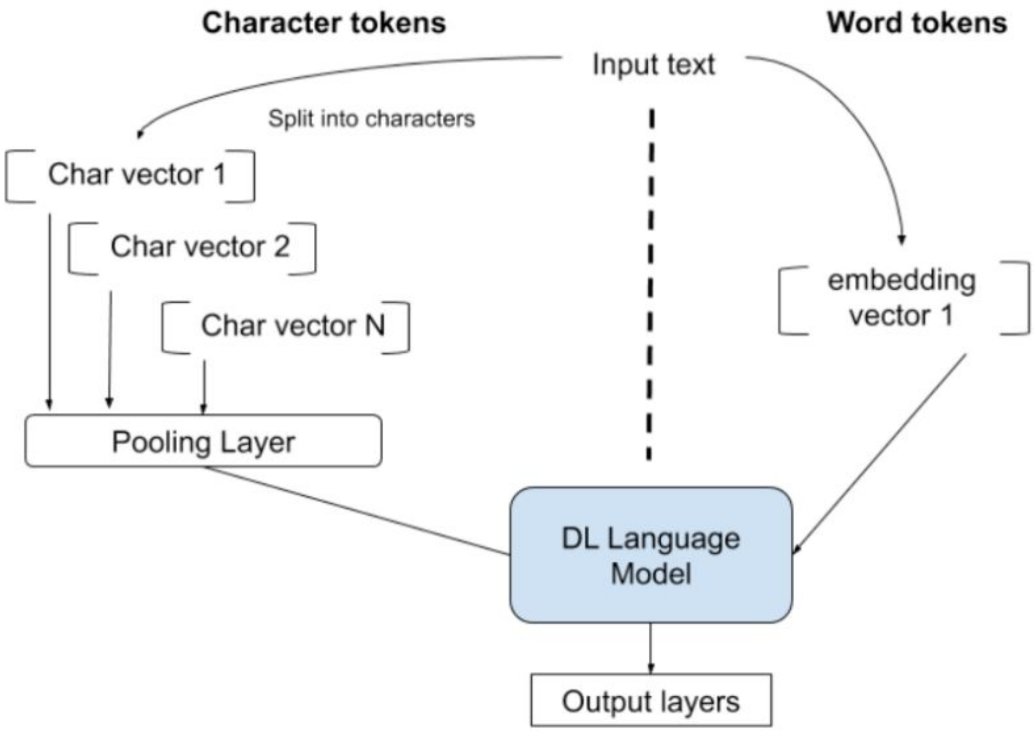
- Pros: Handle unknown or rare words
- Cons:

  Lack of meaning

  Increased input computation

  Limits network choices

Word vs Character Segmentations



Subword Segmentations (Tokenization)

Objective: handle OOV problems with finite vocabulary

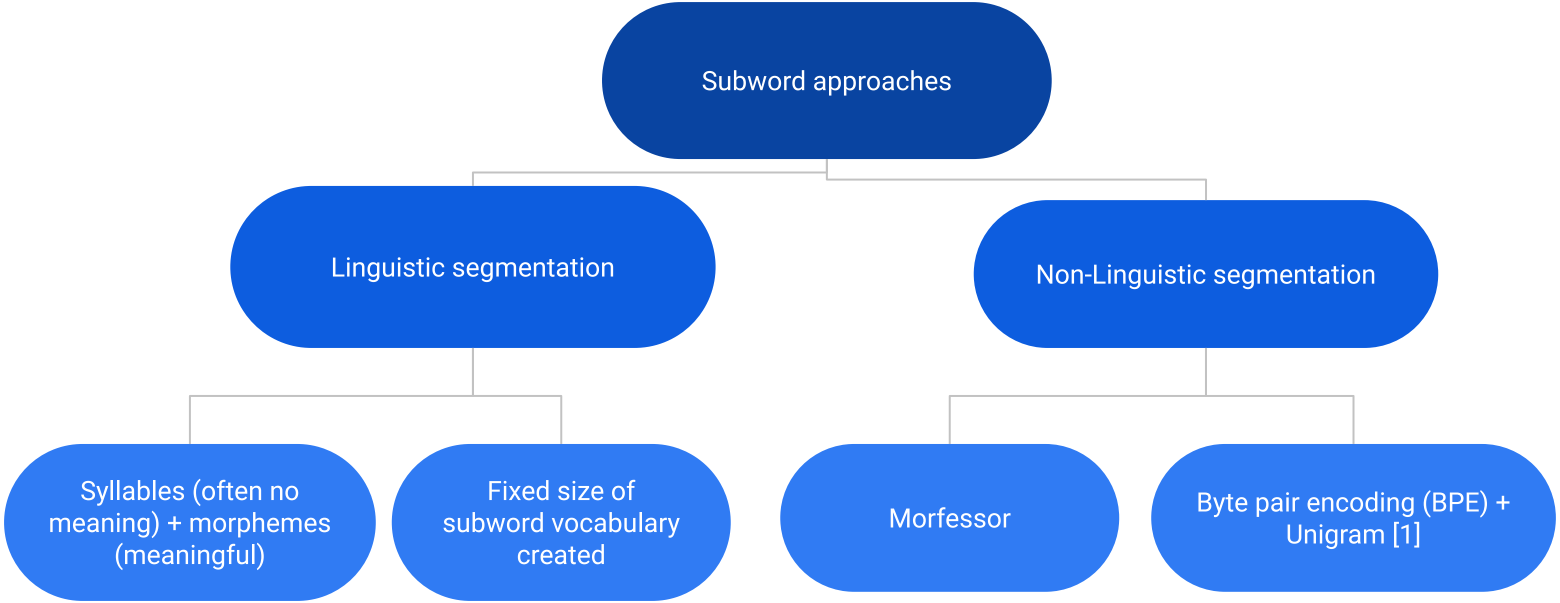
Example:

+ “any” + “place” = “anyplace”
+ “any” + “how” = “anyhow”
+ “any” + “body” = “anybody”

Efficient subword chunks:

“Unfortunately” = “un” + “for” + “tun” + “ate” + “ly”

Subword Segmentation approaches



Morfessor tool

- Probabilistic generative models and tool -> create segmentation models
- words=compounds -> segmented into construction (morphs) + atoms (letter)
- Plan to use
- Example:
  - morfessor-train -s train.bin train.txt
  - morfessor-segment -l train.bin train.txt
  - These command first build the morfessor model “train” and use that model to segment the text file train.txt
  - Compounds: kaksi brittiläistä -> kaksi_ brittiläis tä_

SentencePiece Tool

- Treat whitespace“ “ as “_” symbol
- Lossless Tokenization
- Skip Word Segmentation
- Integer Mapping
- Example:
  - spm_train --input=stt.train.txt.utf8 --model_prefix=bpe --vocab_size=8000 --model_type=bpe
  - spm_encode --model=bpe.model --nbest_size=-1 --alpha=0.5 stt.eval.txt.utf8 --output=bpe_eval.txt
  - Bpe algorithm: kaksi brittiläistä -> _kaksi _brittiläistä
  - Greedy unigram algorithm: kaksi brittiläistä -> _kaksi _brittiläi stä

Evaluation metrics

- Extrinsic evaluation
  - Measures the performance of actual tasks
  - How well the model did on some task (e.g. WER)
  - e.g. speech recognition, spelling corrector, machine translation...
- Intrinsic Evaluation
  - Measures some type of “internal” features of the model
  - e.g. perplexity, cross entropy...

Perplexity (1/2)

- Perplexity is an intrinsic metric of a language model
- Can be thought of as a one kind of a game
  - How well can the language model predict the next word?
- A good language model should be confident

  (and correct)

I would like to drive a ________!

car	(0.1)
bike	(0.01)
…	
tomato	(1e-9)

Perplexity (2/2)

- Perplexity is the probability (of a test set) normalized by the number of words
    - If P(...) ≈ 1, then perplexity is low. Otherwise, perplexity is high.
    - Normalized by the word count because a lot of words makes the probability lower by definition

$$\mathrm{PP}(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

Cross Entropy

- Another intrinsic metric for evaluating a language model
- Measures a difference between two probability distributions

	cat	dog	puppy	snake	rabbit
True distribution	0%	0%	100%	0%	0%
Model distribution	11%	16%	60%	2%	11%

$$\mathrm{H}(p, q) = -\sum_i p_i \log_2(q_i) = -\frac{1}{N} \log_2 \mathrm{P}(w_1, w_2, \ldots, w_N)$$

Experiments (Finnish) (1/2)

- Preliminary experiments with subword algorithms
- Setup
    - 1.5M sentences for training, ~800k unique words
    - 10K sentences for evaluation, ~32k unique words
- Experiment steps
    - Use Morfessor or SentencePiece (BPE, greedy unigram) for subwords
    - Train {1, 2, 3}-gram language models
    - Evaluate OOV and perplexity

Experiments (Finnish) (2/2)

- Findings
  - Vocabulary size significantly lower as expected in comparison with word-based data
  - Perplexity gets quite low as well for 2-gram and 3-gram models
  - OOV is zero for every *n*

Algorithm	Vocab size	1-gram		2-gram		3-gram	
		OOV	PPL	OOV	PPL	OOV	PPL
Morfessor	10707	0	2389	0	368	0	237
BPE	10707	0	2979	0	454	0	271
Greedy unigram	10707	0	1699	0	382	0	226

Next steps

- Try different subword algorithms and vocabulary sizes
  - Effect of the vocabulary size is important to evaluate
- Train a couple of language models using the subword tokens
  - For evaluation, we should train the same model with word-based tokens
  - We aim for having one DL based model as well
- Evaluate the language models on a downstream task
  - We’ll use an speech recognition as the downstream evaluation task

Conclusion

- Subwords can be used in place of regular word-based tokenization
  - Their primary goal is to help with vocabulary issues (e.g. OOV)
- Tools for subwords: BPE, Morfessor and SentencePiece
- Two ways to evaluate
  - Extrinsic: actual performance on a task
  - Intrinsic: e.g. perplexity
- Just a fancy way to split text into tokens

References

[1] Kudo, Taku. 2018. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." arXiv Preprint arXiv:1804.10959.
[2] Kudo, Taku, and John Richardson. 2018. "Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing." arXiv Preprint arXiv:1808.06226.
[3] Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2015. "Neural Machine Translation of Rare Words with Subword Units." arXiv Preprint arXiv:1508.07909.
[4] Iacobelli, F. Perplexity (2015) YouTube
[5] Lascarides, A. Language Models: Evaluation and Smoothing (2020). Foundations of Natural Language Processing (Lecture slides)
[6] Mao, L. Entropy, Perplexity and Its Applications (2019). Lei Mao's Log Book
[7] Peter, Smit. Modern subword- based models for automatic speech recognition.

Q&A Time

Thank you for your participation!