WordPiece

Espen Bang og Iver Omholt

Hva er WordPiece?

- WordPiece er en algoritme for å tokenisere tekst ved å dele ord opp i mindre biter (subwords)
- Opprinnelig utviklet hos Google (for machine translations, speech recognition)
- Brukes i liten grad i klassiske IR systemer. Mer vanlig i moderne IR systemer med NLP.
- Bidirectional Encoder Representations from Transformers (BERT)
- Likheter med Byte Pair Encoding (BPE)

Hva er formålet med WordPiece?

Addressere noen av begrensningene ved standard ordbasert tokenisering. Spesielt for NLP modeller.

- Whitespace, Rule-Based, N-gram tokenisering.
- Regelbasert vs. sannsynlighet

Ønsker å finne en balanse mellom å representere tekst med hele ord, og å dele ord opp i mindre deler når det trengs.

Hva oppnås med WordPiece?

Out-of-Vocabulary words

- Ved å dele opp ord i mindre biter kan man enklere behandle nye ord.
- NLP modeller må forstå ny tekst, feilstavelser, sjeldne eller sammensatte ord.

Redusere størrelse på vokabular

- Mer effektiv trening av maskinlæringsmodeller.

Generalisering

- Ved å gjenkjenne sub-words kan man generalisere bedre på tvers av ordformer

Fleksibilitet på tvers av språk

- Tysk, norsk, finsk

Eksempel

- happiness, joyfulness → happi, joyful, ness
- skrivebord, skrivebok, lesebord → skrive, bord, bok, lese

Algoritme

Corpus:

«all squares are rectangles, but not all rectangles are squares»

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

a, ##l, ##l

r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s

a, ##r, ##e

s, ##q, ##u, ##a, ##r, ##e, ##s

b, ##u, ##t

n, ##o, ##t

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

a, ##l, ##l
r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s
a, ##r, ##e
s, ##q, ##u, ##a, ##r, ##e, ##s
b, ##u, ##t
n, ##o, ##t

Vocabulary:

a, ##l, s, ##q, ##u, ##a, ##r, ##e, ##s, r, ##c, ##t, ##n, ##g, b, n, ##o

Score = freq(AB)/(freq(A)*freq(B))

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

a, ##l, ##l
r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s
a, ##r, ##e
s, ##q, ##u, ##a, ##r, ##e, ##s
b, ##u, ##t
n, ##o, ##t

Vocabulary:

a, ##l, r, ##e, ##c, ##t, ##a, ##n, ##g, ##s, s, ##q, ##u, ##r, b, n, ##o

Score = freq(a, ##l)/(freq(a)*freq(##l))

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

a, ##l, ##l
r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s
a, ##r, ##e
s, ##q, ##u, ##a, ##r, ##e, ##s
b, ##u, ##t
n, ##o, ##t

Vocabulary:

a, ##l, r, ##e, ##c, ##t, ##a, ##n, ##g, ##s, s, ##q, ##u, ##r, b, n, ##o

Score = 2/(4*6) = 2/24 = 0.0833

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

a, ##l, ##l
r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s
a, ##r, ##e
s, ##q, ##u, ##a, ##r, ##e, ##s
b, ##u, ##t
n, ##o, ##t

Vocabulary:

a, ##l, r, ##e, ##c, ##t, ##a, ##n, ##g, ##s, s, ##q, ##u, ##r, b, n, ##o

Algoritme

Corpus:

- all: 2
- rectangles: 2
- are: 2
- squares: 2
- but: 1
- not: 1

Splits:

al, ##l

r, ##e, ##c, ##t, ##a, ##n, ##g, ##l, ##e, ##s

a, ##r, ##e

s, ##q, ##u, ##a, ##r, ##e, ##s

b, ##u, ##t

n, ##o, ##t

Vocabulary:

a, ##l, r, ##e, ##c, ##t, ##a, ##n, ##g, ##s, s, ##q, ##u, ##r, b, n, ##o, al

Algoritme

Vocabulary:

..., all, rectangle, are, square, but, not, ##es, #s, ...

Tokenisering

- notes = not ##es

Vocabulary:

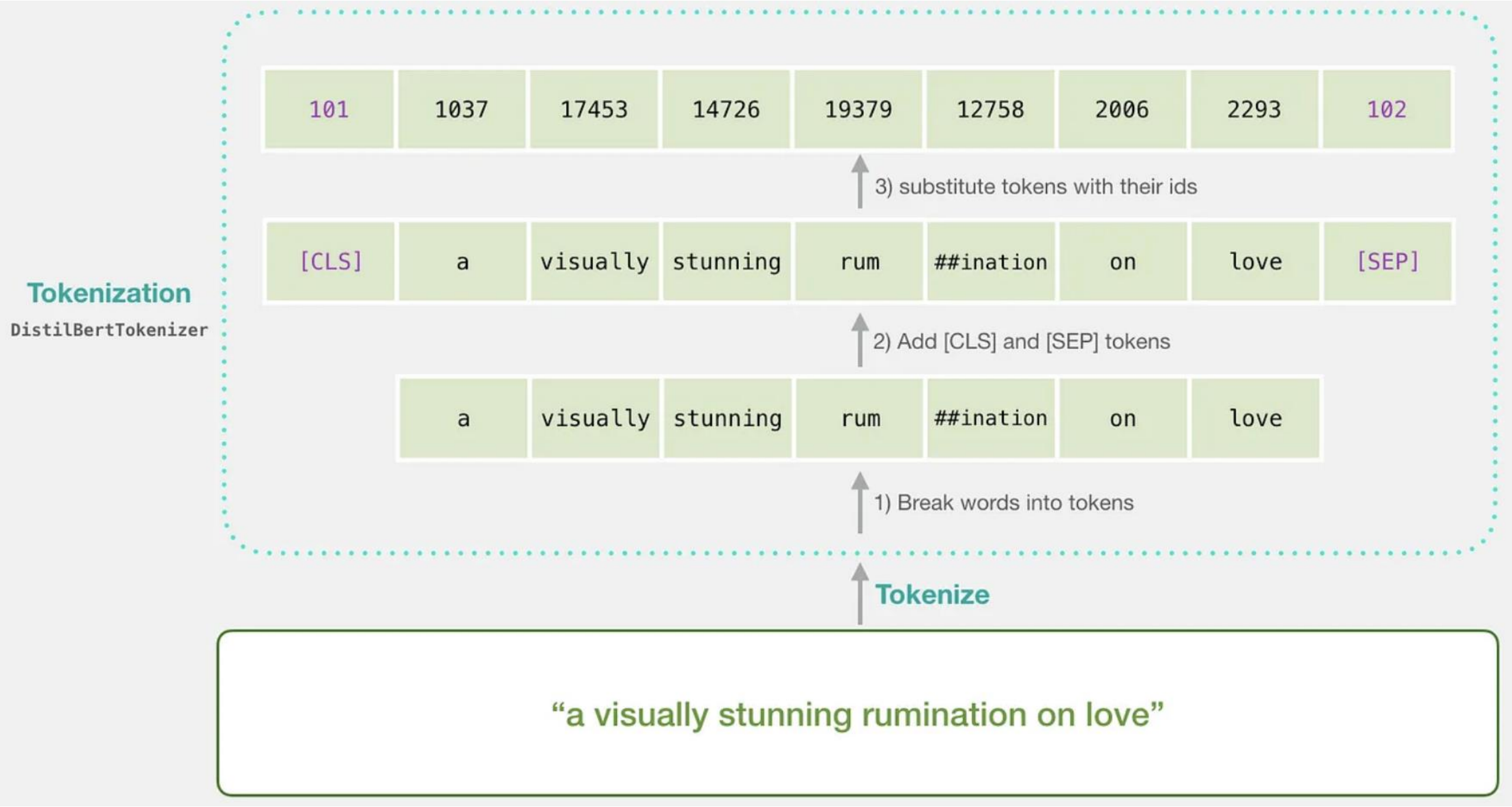
..., all, rectangle, are, square, but, not, #es, #s, ...

Tokenisering

- apples = ?? = [UNK]

Vocabulary:

..., all, rectangle, are, square, but, not, #es, #s, ...


Tokenization
DistilBertTokenizer
101 1037 17453 14726 19379 12758 2006 2293 102
3) substitute tokens with their ids
[CLS] a visually stunning rum ##ination on love [SEP]
2) Add [CLS] and [SEP] tokens
a visually stunning rum ##ination on love
1) Break words into tokens
Tokenize
“a visually stunning rumination on love”