Subword Semantic Hashing for Intent Classification on Small Datasets

Kumar Shridhar et al., arXiv, Dec 2018

Presenter: Gwenaelle Cunha Sergio

Artificial Brain Research Lab., School of Electronics Engineering,
Kyungpook National University

17-Jul-2019

Introduction

- Current models
  - Word embeddings: Word2Vec, GloVe, FastText, ELMo
  - Sentence embeddings: Bag of Words, Skip-thought vectors
  - **Limitations**: vocabulary dependent (Out-Of-Vocabulary issue)

- Challenges
  - *Small dataset* vs *Data-hungry* deep learning methods
  - Dataset: obtained from internet communication (**OOV** and **spelling errors**)

- Focus of this paper
  - Embedding: semantic hashing
  - Obtains rich features to be classified with an intent classifier for small datasets

Semantic Hashing

Method to hash input tokens

"so that the model depends on a hash value rather than on tokens"

Hash: *Transformation of a string of characters into a number key according to a hash function (mathematics manipulation)*

Pre-hashing Algorithm

Algorithm 1 Subword Semantic Hashing

data ← collection of examples
Create **set** *all-sub-tokens*
Create **list** *examples*
for text T in *data* **do**
  Create **list** *example*
  tokens ← split T into words.
  for word w in *tokens* **do**
    w ← "#" + w + "#"
    for x in *length*(w)−2 **do**
      Append $w[x{:}x+2]$ to *all-sub-tokens*
      Append $w[x{:}x+2]$ to *example*
    end for
  end for
  Append *example* to *examples*
end for
return (*all-sub-tokens*, *examples*)

Extraction of trigrams

Pre-hashing function **H**

data = {"change subject line",
"I like flying disk",
"I like strawberries",
...}

T = "change subject line"

tokens = ["change", "subject", "line"]

w = "change"

w = "#change#"

H(change) = ["#ch", "cha", "han", "ang", "nge", "ge#"]

examples = {["#ch", "cha", "han", "ang", "nge", "ge#"],
["#...",
...}

N-gram Encoding

Algorithm 3: N-gram encoding

Data: *alphabet*, *alphabet_vec*, *examples*, *n_size*
Result: Encoded vectors
sentence ← *examples*[*j*];
sentence_ngram ← [];
for *il in len(sentence)* **do**
    s_vec ← *alphabet_vec*[*sentence*[*il*]];
    for *i=1:n_size-1* **do**
        # Elementwise multiplication;
        s_elem ← *alphabet_vec*[*il* + *i*] ∗ *i*;
        # Rotation cia cyclic shift;
        s_vec ← *s_vec*∗ np.roll(*s_elem*);
    sentence_ngram ← *sentence_ngram* + *s_vec*;
sentence_ngram_norm ← norm(*sentence_ngram*);

vectorization
char -> 1x1000 vector

Hash function
in trigram

Sentence embedding

Current corpus

examples = {"#ch cha han ang nge ge#",
"#...",
...}

Parameters

N = 1000 (dimension of desired feature vector)

n_size = 3 (trigram)

alphabet = 'abcdefghijklmnopqrstuvwxyz#' (M = 27)

alphabet_vec = dictionary of size MxN, vectorized version of all characters in *alphabet* with each feature vector being randomly composed of -1s and 1s

Deduced from author's code, not provided in original paper.

Datasets

Chatbot

Intent	Train	Test
Departure Time	43	35
Find Connection	57	71

2 intents

Ask Ubuntu

Intent	Train	Test
Make Update	10	37
Setup Printer	10	13
Shutdown Computer	13	14
Software Recommendation	17	40
None	3	5

5 intents

Web Applications

Intent	Train	Test
Change Password	2	6
Delete Account	7	10
Download Video	1	0
Export Data	2	3
Filter Spam	6	14
Find Alternative	7	16
Sync Accounts	3	6
None	2	4

8 intents

Dataset: https://github.com/sebischair/NLU-Evaluation-Corpora

Results

Platform	Chatbot	AskUbuntu	WebApp	Overall	Avg.
Botfuel	0.98	0.90	0.80	0.91	0.89
Luis	0.98	0.90	0.81	0.91	0.90
Dialogflow	0.93	0.85	0.80	0.87	0.86
Watson	0.97	**0.92**	**0.83**	0.91	**0.91**
Rasa	0.98	0.86	0.74	0.88	0.86
Snips	0.96	0.83	0.78	0.89	0.86
Recast	**0.99**	0.86	0.75	0.89	0.87
TildeCNN	**0.99**	**0.92**	0.81	**0.92**	**0.91**
Our Avg.	0.98	**0.92**	**0.83**	**0.92**	**0.91**
Our Best	**0.99**	**0.93**	**0.85**	**0.93**	**0.92**
Best Ind.	*1.00*	*0.93*	*0.86*	*0.94*	*0.93*

Thank you!