Created By: Aakash Goel
Objective: Intent Classification using HINT3 dataset (sofmattress)

Created on: 26 - May - 2021
Last updated on: 27 - May - 2021

Table of Contents

1. Setting up problem

Given a user typing free text (query on chatbot), find its intent among following:

EMI, COD, ORTHO_FEATURES, ERGO_FEATURES, COMPARISON, WARRANTY, 100_NIGHT_TRIAL_OFFER, SIZE_CUSTOMIZATION, WHAT_SIZE_TO_ORDER, LEAD_GEN, CHECK_PINCODE, DISTRIBUTORS, MATTRESS_COST, PRODUCT_VARIANTS, ABOUT_SOF_MATTRESS, DELAY_IN_DELIVERY, ORDER_STATUS, RETURN_EXCHANGE, CANCEL_ORDER, PILLOWS, OFFERS, NO_NODES_DETECTED

Identifying the broad category that the users are trying to find will narrow down the possible results.
Sample dummy input and output:
Input: `"What about size"`
Output: WHAT_SIZE_TO_ORDER

2. Data

2.1. Training data

2.1.1. Sentence and Label

	sentence	label
0	You guys provide EMI option?	EMI
1	Do you offer Zero Percent EMI payment options?	EMI
2	0% EMI.	EMI
3	EMI	EMI
4	I want in installment	EMI

External Data

Get a description of Label if possible using some external database like DbPedia to increase labelled data.

2.1.2. Semantic Embeddings (Google + Facebook -- Sub Word Level) + Contextualised Embeddings - ELMO + Sentence Level Embedding -- USE (Google Universal Sentence Encoder)

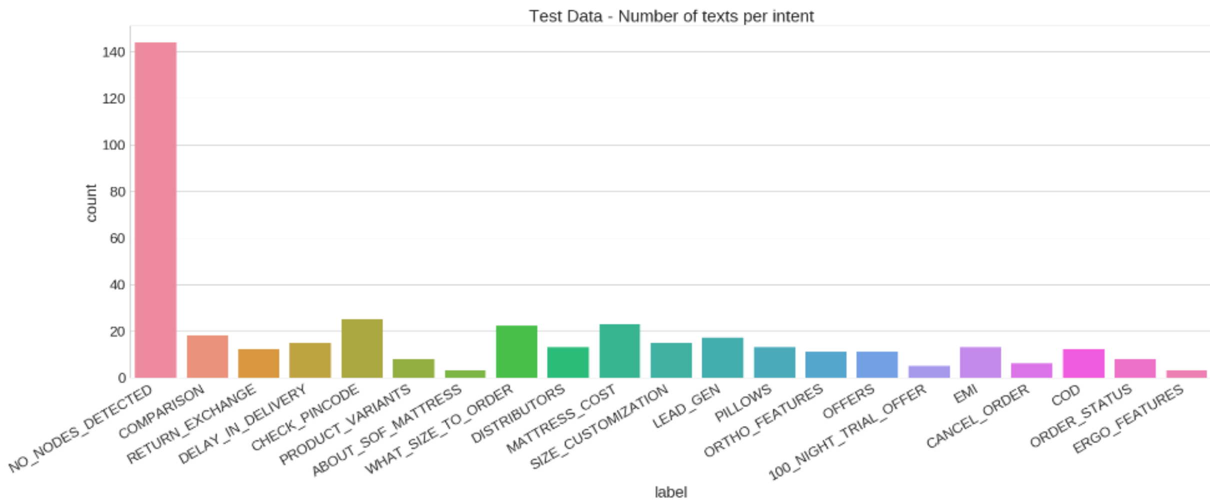
Objective is to get an embedding of the corpus. Will use python gensim's word2vec package to train embeddings.

2.1.3. Spelling Correction and Word Segmentation

Edit Distance based approach.

2.2. Test data

	sentence	label
0	There are only 2 models	NO_NODES_DETECTED
1	Single	NO_NODES_DETECTED
2	What's difference between ergo and ortho	COMPARISON
3	Return order	RETURN_EXCHANGE
4	Hai not recieved my product	DELAY_IN_DELIVERY



'NO_NODES_DETECTED' label present in Test data but not in Training data.

3. Solution

Input strings pass through the Query Preprocessing Module and then by Intent classification Module.

3.1. Query Preprocessing

3.1.1. Basic Pre-processing step - Optional (Depend on Algorithm using)

3.1.1.1. Case conversion, remove punctuations, stopwords, multiple spaces between tokens

3.1.2. Word Segmentation - Optional (Depend on Algorithm using)

Segmentation module splits string into two parts while iterating through string, character at a time and generates possible candidate words. So, there are n–1 positions between characters, each of which can either be or not be a word boundary Iterate through string. Using the Unigram and Bi-gram model, it chooses the best possible candidate.

3.1.3. Spelling Correction - Optional (Depend on Algorithm using but preferable)

3.1.3.1. Isolated correction

Treat each query word in an isolated manner while doing correction. Now, let's see the detailed **methodology** behind **Isolated correction**.
Let say, the query contains *Q = {w1 w2}* and needs to find the most likely spelling correction for *Q i.e {C_w1 C_w2}* where *C_w1 and C_w2* is corrected spelling for w1 and w2 respectively. Given the original word *w*, we need to choose best among possible corrections by taking *argmax of P(c/w)* where *c* belongs to *C* (list of candidate words). Using Bayes' theorem, *argmax of P(c/w)* equivalent to *P(c)*P(w/c)/P(w)*, ignore *P(w)* as it is the same for all candidate words.

3.1.3.1.1. Generate candidate words

Using edit distance based on 4 operations delete (remove letter), transpose (swap adjacent letters), replace (change one letter to another), insert (add letter). Let's take 2 edit distances as of now.

3.1.3.1.2. Probability of candidate *P(c)*

Probability that *c* appears in our corpus. Example - *P(juice)* is equal to the ratio of the number of times juice appears in the corpus to the sum of frequency of all words in the corpus.

3.1.3.1.3. Error Model *P(w/c)*

Probability that w would be typed when the user meant c. Example - P(juse/juice) should be relatively higher than P(juse/just).

3.1.3.2. Context sensitive correction

It considers each query word to correct the complete query. Context plays a role while query correction like even if all words are correct in the query still it can be mis-spelled.
The probability of a sequence of words is the product of the probabilities of each word, given the word's context: all the preceding words $P(W_{1:n}) = \prod_{k=1}^{n} P(W_k \mid W_{1:k-1})$.
Please see below screenshot from swiggy search

3.2. Intent Classification

Model/Learn P(class = Label/ query).

3.2.1. Feature Engineering

3.2.1.1. Bag of Words & syntactic

Represent Query as a bag of words (Tf-IDF Vectorizer), try with different levels of n-grams (character, word, length). In addition, also add below features

3.2.1.1.1. Number of character in query

3.2.1.1.2. Number of words in query, Numerical digits

3.2.1.2. Embedding Vector

- Represent Query as fixed Vector size using embeddings trained on training data. Check 2-D visualisations over learned embeddings to validate quality of embeddings.
- Sentence Level Embedding using Google Universal Sentence Encoder
- Use Pre-trained Language Models, Transformer - BERT.

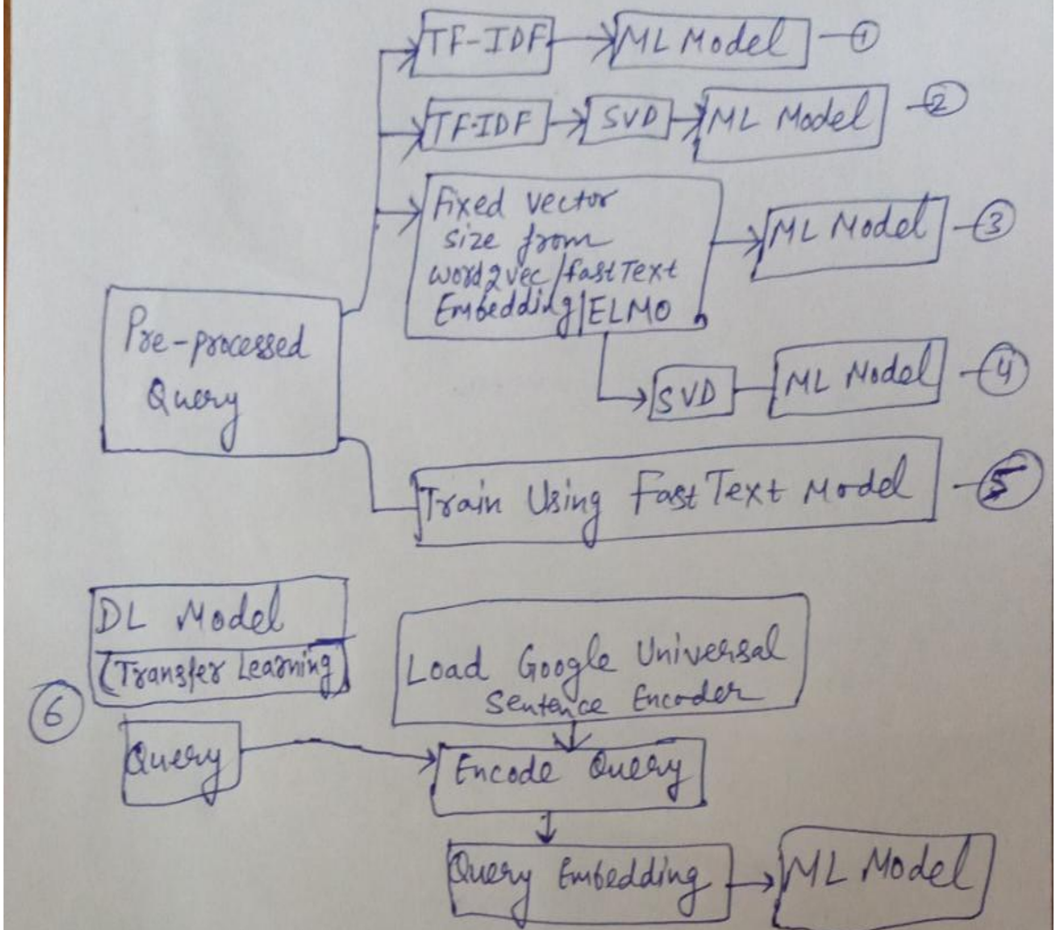
3.2.2. Algorithm (Fine Tuning Pre-trained Model - Transfer Learning)

3.2.2.1. Supervised

Experiment (Try and check performance) following classifiers on DTM (document term matrix) obtained from "*feature engineering*" step - 3.2.1.1 & 3.2.1.2 separately. Below figure contains different ways in which supervised models can be tried out.

Created on: 27-May-21
By: AAKASH GOEL

ML Model



(7) fine Tune Bert Model (Add Dense Layer, softmax)

(8) ULMFIT → Fast AI

Keyword based Approach (Not scalable)

* use Different lexicon sources to enrich list of seed words for each class/Label.
* Develop Scoring Model.

SVD - Singular Value Decomposition
ML Model - Refer to any of below model

3.2.2.1.1. Naive Bayes

Apply Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

$$\boxed{P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}}$$

Y = Output variable
X = Feature from DTM

3.2.2.1.2. Logistic Regression

Linear model for classification, probabilities describing the possible outcomes of a single trial are modeled using a logistic function. Minimize following cost function

$$h_\theta(x) = \frac{1}{1 + e^{\theta^\top x}}$$

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] \end{aligned}$$

3.2.2.1.3. Linear Support Vector classification

It constructs a hyper-plane high dimensional space, which can be used for classification. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (functional margin), since in general the larger the margin the lower the generalization error of the classifier. Two ways to measure model: how many misclassified points and how wide is margin (Error = C* classification error + Margin Error)

3.2.2.1.4. <u>FastText Supervised classification Model</u>

Quick and easy to train. Use a Multi Layer Neural Network with 10 hidden units.

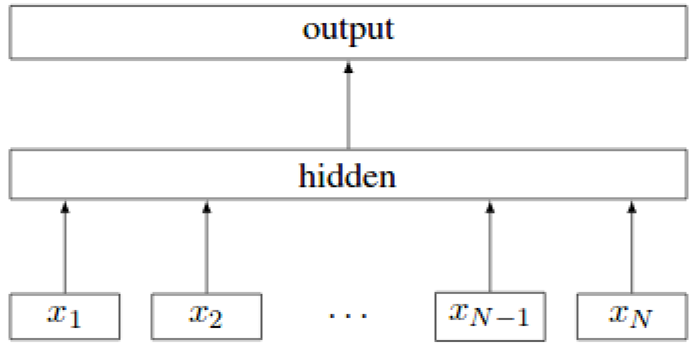
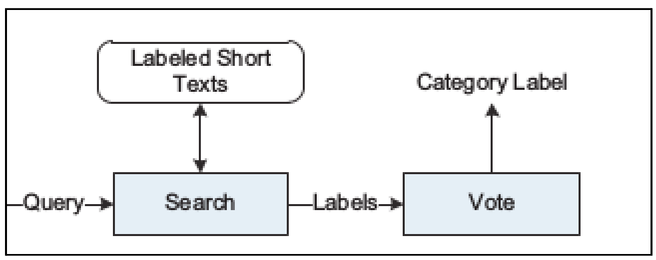


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

3.2.2.2. <u>Semi-supervised</u>

3.2.2.2.1. <u>Query Partial and Complete Match in Lexicon of Label</u>

As chances of query length being large is very less, do partial and complete match of query with seed words of each label and then select label using voting for each label.



3.2.2.2.2. <u>Cosine Similarity (Query and Lexicon of Label)</u>

As we already trained embeddings on our data, find cosine similarity between Query and each candidate name (Label) and take **Best-K scores** and apply **voting** on those best scorer labels.

4. Benchmark results

Below are benchmark results and SOTA for intent classification by different research groups:

4.1. Botfuel

corpus	num of intents	train	test
Chatbot	2	100	106
Ask Ubuntu	5	53	109
Web Applications	8	30	59

Intent classification results

While the paper did the benchmark for both intent classification and entity extraction, we will focus only on intent classification. We compute the `f1` score for each corpus and the overall `f1`:

Platform\Corpus	Chatbot	Ask Ubuntu	Web Applications	Overall
Botfuel	0.98	0.90	0.80	0.91
Luis	0.98	0.90	0.81	0.91
API (DialogFlow)	0.93	0.85	0.80	0.87
Watson	0.97	0.92	0.83	0.92
RASA	0.98	0.86	0.74	0.88
Snips	0.96	0.83	0.78	0.89
Recast	0.99	0.86	0.75	0.89

4.2. HINT3

	SOFMattress		Curekart		Powerplay11	
	Full	Subset	Full	Subset	Full	Subset
Dialogflow	73.1	**65.3**	75.0	71.2	59.6	55.6
RASA	69.2	56.2	**84.0**	80.5	49.0	38.5
LUIS	59.3	49.3	72.5	71.6	48.0	44.0
Haptik	72.2	64.0	80.3	79.8	**66.5**	**59.2**
BERT	**73.5**	57.1	83.6	**82.3**	58.5	53.0

Table 3: Inscope Accuracy at low threshold=0.1 for Full and Subset data variants

5. References

5.1. https://norvig.com/spell-correct.html
5.2. Efficient Intent Detection with Dual Sentence Encoders -- https://arxiv.org/pdf/2003.04807.pdf
5.3. https://www.aclweb.org/anthology/2020.insights-1.16.pdf
5.4. https://link.medium.com/Mm4pqGnLzgb
5.5. Universal Sentence Encoders -- https://arxiv.org/abs/1803.11175
5.6. ConveRT -- https://arxiv.org/pdf/1911.03688.pdf
5.7. Bag of Tricks for Efficient Text Classification -- https://www.aclweb.org/anthology/E17-2068.pdf
5.8. ttps://fasttext.cc/docs/en/python-module.html#text-classification-model
5.9. https://www.researchgate.net/profile/Dennis_Mazur/post/NLP_Short_text_classification_categorization/attachment/59d6583579197b80779ae38b/AS%3A537608355880960%401505187228734/download/cdee73241094c09b6ca0b37dd1c8cd4ddd7d.pdf
5.10. http://personales.upv.es/prosso/resources/HernandezEtAl_CERI12.pdf
5.11. https://github.com/Botfuel/benchmark-nlp-2018
5.12. https://medium.com/snips-ai/benchmarking-natural-language-understanding-systems-google-facebook-microsoft-and-snips-2b8ddcf9fb19