DEPARTMENT OF INFORMATICS

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Master's Thesis in Data Science

Using sentence embeddings for the management of user-generated training phrases in Natural Language Understanding (NLU) service

Author:	Apostolos Tamvakis
Academic Supervisor:	Prodromos Malakasiotis
Capstone Project Advisor:	Stavros Vassos, Helvia.io

Abstract

In this thesis, we are tasked to explore possible implementations to develop a mechanism for detecting similar Question Answering sets (intents). This is also referred as Semantic Search Similarity. The goal is to determine which is the best model fitted for this situation and deploy it as a service for Business Intelligence (BI) purposes. Advances in this area, which is a part of a general concept called Natural Language Inference, have been substantial. Latest neural network models combine both unsupervised and supervised knowledge to produce context aware vector representation of documents. The two most prominent models, according to bibliography, Universal Sentence Encoder (USE) and SentenceBERT are evaluated on an annotated dataset provided by Helvia Technologies. The results indicate that these pretrained models are able to detect semantically related sets where better than the baseline models. Finally we introduce a Human in the Loop evaluation system where the company's personnel is asked to test the proposed best models and see if this serves as a useful tool in reducing the processing time of manually merging similar sets.

Περίληψη

Για αυτή την εργασία, εξερευνούμε πιθανές υλοποιήσεις για να αναπτυχθεί ένας μηχανισμός ανίχνευσης παρόμοιων σετ Ερωτήσεων/Απαντήσεων. Αυτή η διαδικασία αναφέρεται ως Σημασιολογική Αναζήτηση Ομοιοτήτων (Semantic Search Similarity). Σκοπός είναι να καθοριστεί το καλύτερο μοντέλο για αυτό το πρόβλημα και να προσφερθεί ως μία υπηρεσία Επιχειρησιακής νοημοσύνης. Η εξέλιξη της τεχνολογίας σε αυτή την περιοχή, που αποτελεί και μέρος ενός γενικότερου προβλήματος στη Επεξεργασία Φυσικής Γλώσσας το οποίο λέγεται Εξαγωγή Συμπεράσματος (Natural Language Inference) , έχει σημειώσει σημαντική πρόοδο. Τα νεώτερα μοντέλα νευρωνικών δικτύων βαθιάς μάθησης συνδυάζουν επιτηρούμενη και και μη επιτηρούμενη μάθηση για να παράγουν διανυσματικές αναπαραστάσεις κειμένων συλλαμβάνοντας σημασιολογικό περιεχόμενο. Τα δύο καλύτερα μοντέλα που προέκυψαν μετά τα πειράματα στα δεδομένα που παραχωρήθηκαν από την Helvia Technologies , θα δοκιμαστούν σε περιβάλλον παραγωγής. Τα δικά μας πειράματα έδειξαν πως τα προεκπαιδευμένα νευρωνικά δίκτυα USE & SentenceBert κατάφεραν να εξάγουν σημασιολογικές ομοιότητες μεταξύ των σετ σε αντίθεση με τα μοντέλα αναφοράς. Τέλος, προτείνεται η αξιολόγηση αυτού του συστήματος μέσω μιας Human in the Loop προσέγγισης όπου οι άνθρωποι αξιολογούν την απόδοση αυτού. Δηλαδή, κατά πόσο αντιστοιχούν στην πραγματικότητα οι προτάσεις του εργαλείου περί όμοιων σετ, όπως επίσης από το κατά πόσο γρηγορότερη γίνεται η διαδικασία ομαδοποίησης.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my academic supervisor, Prodromos Malakasiotis, for his constant guidance and mentoring throughout the course of this thesis. His domain knowledge was truly helpful for the completion of this project.

In addition, I would like to thank my company supervisor, Stavros Vassos, for his constructive collaboration. Sharing useful insights about the inner workings and the future development of the product, were proven very valuable to me.

Contents

Chapter 1

Introduction

Semantic Textual Similarity (STS) determines the level of semantic similarity between two texts. In their majority, STS systems calculate the similarity score, according to a metric, on a fixed scale. With the advance of deep learning research, newer and more effective models have come to assist in the continuous challenges presented in the STS tasks.

1.1 Problem Statement

This thesis,which serves also as a capstone project with Helvia Technologies, aims to explore how a variety of Natural Language Processing Systems (NLP) for the STS task can be featured in a production level design.

More specific, we are trying to identify similar components in user-generated content for Natural Language Understanding (NLU) services. This serves for detecting similar intents that should be merged due to relevant similarity in their user-generated training phrases (questions/answers).

To solve this, we are reviewing the standard techniques fit to problem's characteristics and the latest state-of-the-art, referred in the literature.

1.2 Outline

The rest of the thesis is organized as follows:

- Chapter 2: Review of the related work.
- Chapter 3: Presentation of the methodology followed.
- Chapter 4: Experimental Results.

- Chapter 5: Synopsis of the thesis and future development.

Chapter 2

Related Work

2.1 Text vectorization

Machine learning algorithms operate on a numeric feature space, expecting input as a two-dimensional array where rows are instances and columns are features. In NLP , we need to transform the documents into a vector form in order to apply a certain ML algorithm. This process is called feature extraction or vectorization, and is an essential first step toward language-aware analysis.

2.1.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is the most commonly used method in NLP and Information Retrieval for converting text documents into matrix representation of vectors. The TF representation weights terms t proportionally to their frequency in a corpus. The IDF representation penalizes common tokens and rewards rare tokens in the vector representation.

$$\begin{aligned} \text{TF}_{(t,d)} &= \frac{\text{count}(t,d)}{\text{total words in d}} \\ \text{IDF}_t &= log(\frac{N}{DF+1}) \\ \text{TF-IDF}_{(t,d)} &= \text{TF}_{(t,d)} * \text{IDF}_t \end{aligned}$$

where:
count(t, d) is the number of occurrences of t in document d,
DF is the number of documents containing t,
N is the total number of documents in the corpus.

2.2 Unsupervised learning

2.2.1 Word Embeddings

Distributed representations of words (word embeddings) (Bengio et al. [1]) display useful features for various NLP tasks. Each word corresponds to a vector in an feature space, where words with similar meaning have a short distance. Estimation of the word vectors itself was performed using different neural network model architectures and trained on various corpora (Google News,Wikipedia). While the there is a common approval for the usefulness of word representations, it is yet troublesome that can not fully represent the meaning in a complete sentence. This due to the fact that the order of words in the sentence is not taken into account.

The most notable word embedding models are Word2Vec by Mikolov et al. [2] and GloVe by Pennigton et al. [3]. Since these models are trained on unsupervised data, it is necessary to create a "pseudo" task for the network to train. The main goal is for the network to learn the weights of the hidden layer that represent the word vectors.

Word2Vec has two variations:

Skip-Gram

In Skip-Gram the task is, to predict the neighboring words of a given word within a window.

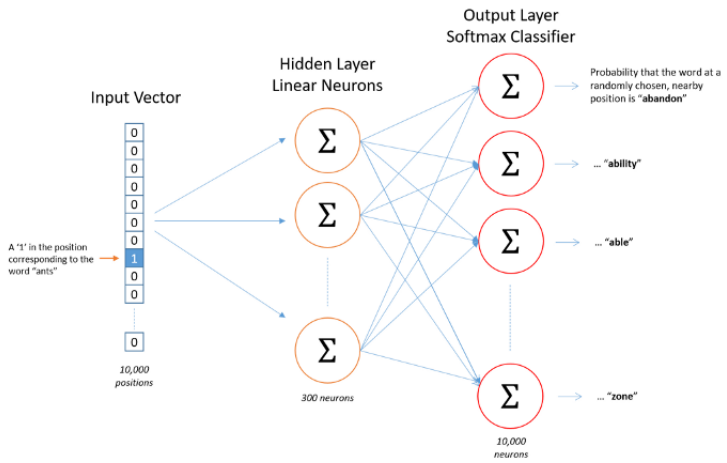

Figure 2.1: Architecture for skip-gram model.

CBOW

In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction

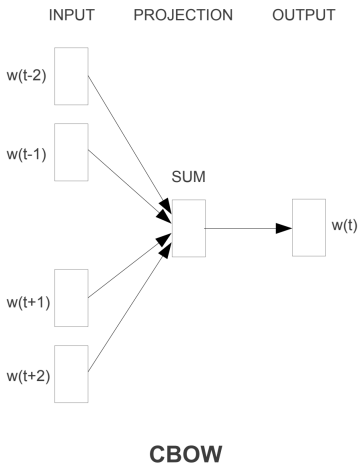

Figure 2.2: Architecture of the CBOW

Weighted Word vectors

In this method,we are weighting the word vectors with their respective TF-IDF score and then averaging the word embeddings to create a centroid representation for the sentence. This method serves also as a baseline for NLP tasks and is presented in [4],[5]

$$\vec{t} = \frac{\sum_{j=1}^{|V|} \vec{w_j} * TF(w_j, t) * IDF(wj)}{\sum_{j=1}^{|V|} *TF(w_j, t) * IDF(wj)} \tag{2.1}$$

where,

$\vec{t}$ is a vector representation for text $t = \langle w_1, w_2, \ldots, w_n \rangle$ of n consecutive word occurrences,

$|V|$ is the number of (distinct) words in the vocabulary,

$TF(w_j, t)$, is the term frequency (number of occurrences) of the j-th vocabulary word in the text t,

$\vec{w_j}$ is a vector representation of the j-th word in the vocabulary.

Sentence vectors is an extension to word vector representation where feature representations at sentence level or the complete document are learned instead of words,by averaging the vector representations of all words in the sentence.

2.2.2 Sentence Embeddings

Many neural NLP systems are initialized with pretrained word embeddings but learn their representations of words in context from scratch, in a task-specific manner from supervised learning input.However, learning these representations from the start is not always possible, especially in cases with restricted resources, where it is believed that using general purpose sentence representations will be beneficial.

Skip-Thought Vectors [6] released in 2015 have made good progress in sentence-level embeddings.

Arora et al. [7] proposed a method of sentence embedding called smooth inverse frequency (SIF) as a simple baseline for all sentence representations to overcome. The method involves taking the mean of all word vectors in a list and removing the first principal component. They found that this simple method or sentence embedding creates satisfactory results. The hypothesis is that SIF removes the effect of most common words that occur in documents, and is correlated with removing stop words from consideration. SIF serves as the method of sentence representation tested in all models in this research.

2.3 Supervised Learning

Supervised learning requires labelled data for creating sentence embeddings.The training data used, are annotated for specific tasks like the Stanford Natural Language Inference task [8]. Ideally, the annotated data should be suitable enough for generating accurate sentence vectors.

Conneau et al. [9] explore this concept, studying 7 different sentence encoding architectures into fixed-sized representations. Their research focused on typical recurrent models (LSTMs, GRUs), for which they examined mean and max-pooling over the hidden representations, a self-attentive network that incorporates different views of the sentence, and a hierarchical convolutional network that can be seen as a tree-based method that blends different levels of abstraction. Results were positive, indicating training on NLI problems have better performance against existing unsupervised approaches.

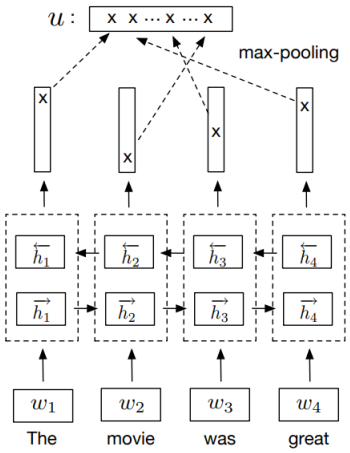

Figure 2: **Bi-LSTM max-pooling network.**

Figure 2.3: InferSent encoder [9]

2.3.1 Universal sentence Encoder

In [10] Cer et al. propose two models for producing sentence embeddings that demonstrate good transfer to a number of other NLP tasks. They experimented with varying amounts of transfer task training data to suggest the relationship between transfer task performance and training set size. Results showed that the sentence embeddings can be used to obtain surprisingly good task performance with remarkably little task specific training data.

USE architecture

The authors offer two variations on the architecture which has different objectives. One based on the transformer architecture targets high accuracy at the cost of greater model complexity and resource consumption. The other targets efficient inference with slightly reduced accuracy.

USE incorporates both supervised and unsupervised (self-supervised learning) training data. The sources are Wikipedia, web news, web question-answer pages and discussion forums which are used for self-supervised learning. In the sequel, they augment them with training on supervised Data from the SNLI corpus.

Transformer

The transformer based sentence encoding model creates sentence embeddings using the encoding sub-graph of the transformer [11]. This sub-graph uses an attention mechanism to compute context aware representations of words in a sentence

that take into account both the ordering and identity of the rest of the words. The context aware word representations are transformed to a standard length sentence encoding vector by computing the element-wise sum of the representations at each word position.The encoder receives as input a lowercased Penn TreeBank (PTB) tokenized string and outputs a 512 dimensional vector.

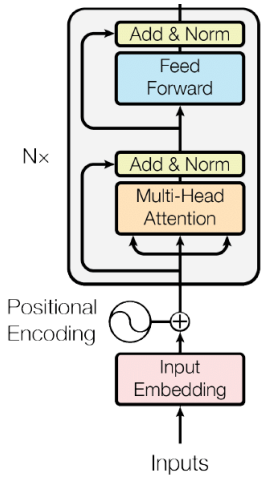

Figure 2.4: Encoder architecture as proposed in [11]

Deep Averaging Networks (DAN)

The second encoding model makes use of a deep averaging network (DAN) [12] in which input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to create the sentence vectors. The network input is identical to the previous method. A significant factor to this approach is that they make use of multitask learning where the encoder is used to provide sentence embeddings for multiple tasks.

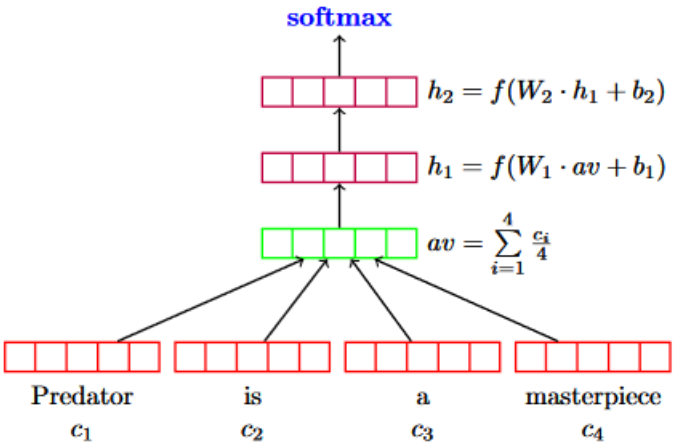

Figure 2.5: DAN architecture [12]

2.3.2 Sentence BERT

SBERT is based on previous works [13] on sentence representation from contextual embeddings. Models such as BERT, can generate different vector representations for the same word in different sentences depending on the surrounding words.

The most common BERT-based methods generate sentence embeddings by simply passing a single sentence and then derive a fixed sized vector by either averaging the outputs (similar to average word embeddings) or by using the output of the special CLS token.

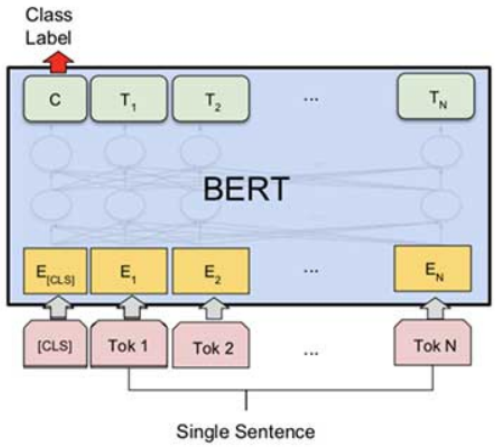

Figure 2.6: Bert Sentence embedding extraction using the CLS token

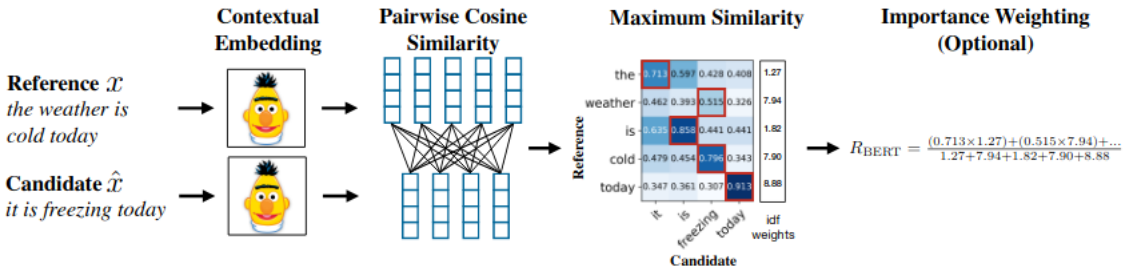

Figure 2.7: Illustration of the computation of the recall metric R_{BERT}. Given the reference x and candidate $\hat{x}$,BERT embeddings and pairwise cosine similarity are estimated [13].

Reimers et al. [14] showed that on textual similarity (STS) tasks, using either the averaging or [CLS] method for sentence embeddings using BERT gives poor results. Even GloVe vectors significantly outperform naive BERT sentence embeddings.

Instead, Reimers et al. present an alternate form of the pretrained BERT network that use siamese and triplet network structures to extract semantically meaningful sentence embeddings using cosine-similarity for comparison and a mean-squared loss function for the inference task.

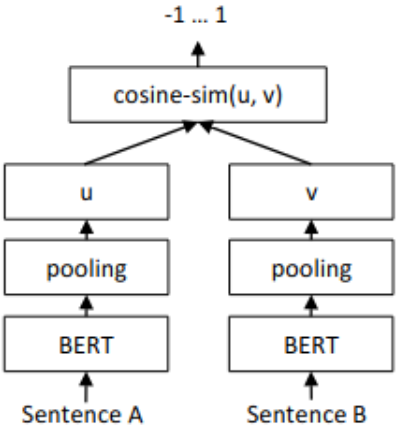

Figure 2.8: SBERT architecture at inference, for example, to compute similarity scores.

SBERT adds a pooling operation to the output of the encoder to extract a fixed sized sentence embedding. They experiment with three pooling strategies: Using the output of the CLS-token,computing the mean of all output vectors (MEAN-strategy,default configuration), and computing a max-over-time of the output vectors (MAX-strategy).

2.4 Benchmarks on Semantic Similarity

The Semantic Textual Similarity (STS) benchmark measures the ability of models to replicate fine-grained graded human judgements of pairwise English sentence similarity. Models are scored according to their Pearson/Spearman correlation, r/ρ, on gold labels ranging from 0, unrelated meaning, to 5, semantically equivalent, with intermediate values capturing carefully defined degrees of meaning overlap. STS is broadly used to evaluate the quality of sentence-level embeddings by assessing the degree to which similarity between pairs of sentence embeddings aligns with human perception of sentence meaning similarity

The SICK dataset, consists of about 10,000 English sentence pairs annotated for relatedness in meaning and entailment.It was built from two existing datasets: the 8K ImageFlickr data set and the SemEval-2012 STS MSR-Video Descriptions dataset.

Model	Spearman
Not trained for STS	
Avg. GloVe embeddings	58.02
Avg. BERT embeddings	46.35
InferSent - GloVe	68.03
Universal Sentence Encoder	74.92
SBERT-NLI-base	77.03
SBERT-NLI-large	79.23
Trained on STS benchmark dataset	
BERT-STSb-base	84.30 ± 0.76
SBERT-STSb-base	84.67 ± 0.19
SRoBERTa-STSb-base	**84.92** ± 0.34
BERT-STSb-large	**85.64** ± 0.81
SBERT-STSb-large	84.45 ± 0.43
SRoBERTa-STSb-large	85.02 ± 0.76
Trained on NLI data + STS benchmark data	
BERT-NLI-STSb-base	**88.33** ± 0.19
SBERT-NLI-STSb-base	85.35 ± 0.17
SRoBERTa-NLI-STSb-base	84.79 ± 0.38
BERT-NLI-STSb-large	**88.77** ± 0.46
SBERT-NLI-STSb-large	86.10 ± 0.13
SRoBERTa-NLI-STSb-large	86.15 ± 0.35

Figure 2.9: Evaluation benchmarks on the STS benchmark test set. The similarity score is estimated as Spearman's rank correlation between the cosine-similarity of the sentence embeddings and the gold labels.

The Microsoft Research Paraphrase Corpus (MRPC) corpus is a paraphrase identification dataset, where systems aim to identify if two sentences are paraphrases of each other. The evaluation metric is classification accuracy and F1.

Chapter 3

Semantic Search for Similar Text Content Using Embeddings

In this chapter, is presented the methodology followed for grouping similar text content performing semantic search. The dataset comes from NLU service provided by Helvia Technologies which offers automated Human Resources assistants. The client provides question/answer (QA) sets for a specific topic (intent). A problem that occurs from this procedure is that many similar user generated QA sets have been labeled with a different intent, something that poses an overhead to the system and the human moderator that has to filter such events. A solution to this problem is proposed in this thesis, by exploring both traditional and state of the art tools for producing embeddings for the QA sets.

3.1 The dataset

The evaluation dataset consists of 105 QA sets and each set includes one or more question/answer phrases.

In total there are:

- 124 question examples
- 105 answer examples

At this point we have to mention that 93 of the 105 answers are unique to their related question.This implies that all questions that share the same answer should be grouped together.

The sets that are merged in groups from the annotator, have been flagged according to a representative index from each group.

The following table presents information about the annotated relevant intents per query.

Relevant Documents per Query	
Average	2.6
St.Deviation	0.966
Max	5
Min	2
Total groups	17

Table 3.1: Information on the relevant intents per query.

3.2 Methodology

In order to create the best model for detecting similar intents, we have to evaluate a selection of models according to the metrics that fit the problem best.

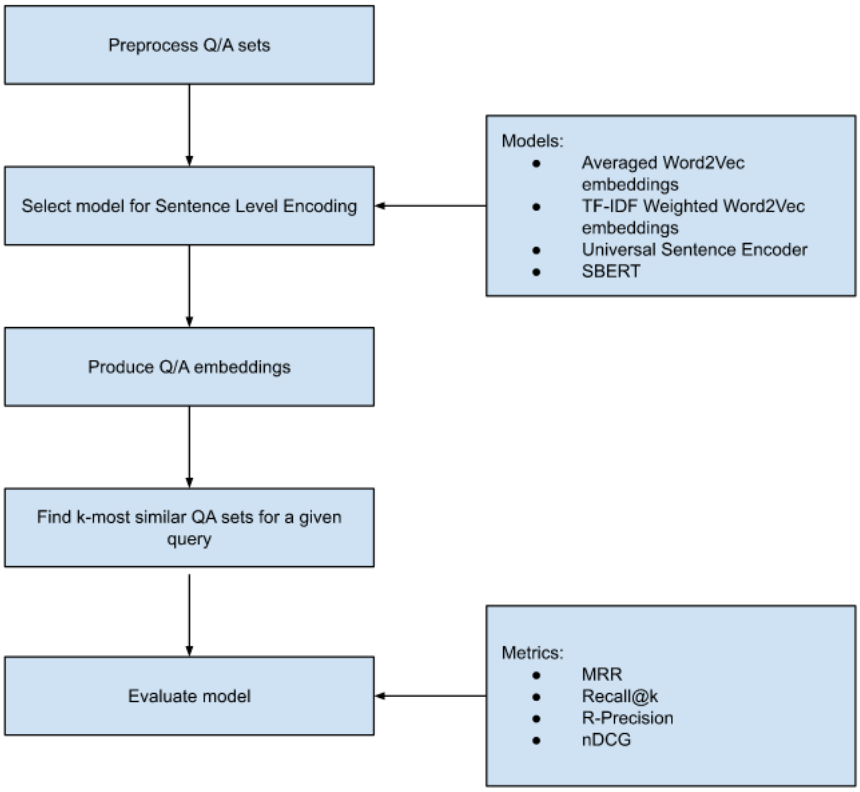

Figure 3.1: Benchmarking Pipeline

3.3 Encoder Selection

To produce a vector representation for a given QA set which may contain multiple examples, first we have to embed each example and then perform a pooling operation

on the intent examples to represent the set as a centroid. The final outcome can have 3 different centroid representations:

- Question centroids
- Answer centroids
- QA centroids (pooling operation on Question,Answer centroids)

3.3.1 Word2Vec model

For this model, we create a vocabulary from the QA answering corpus and for a given sentence we average the vectors corresponding to the words found both on the corpus vocabulary and Word2Vec model vocabulary. Each word vector has a shape of (1,300) therefore each sentence vector will have the shame shape.

3.3.2 TF-IDF weighted Word2Vec

We follow the same technique referred in Equation (2.1) because we may want to shift the weight of the sentence to the most important words. In order for this to be successful, we have to carefully tokenize words in order to match the Word2Vec's vocabulary.

The same tokenizer (nltk's Word Tokenizer) was applied for both models in order to obtain the same vocabulary. In summary:

- For the question examples there are:
  - 373 words matched with Wor2Vec vocabulary
  - 34 found Out of Vocabulary (OOV)
- For the answer examples there are:
  - 872 words matched with Wor2Vec vocabulary
  - 193 found Out of Vocabulary (OOV)

The form of the data poses a difficult challenge for the models since it contains multiple html addresses, abbreviations and content specific definitions.

3.3.3 USE encoder

The model is trained and optimized for greater-than-word length text. The input is variable length English text and the output is a 512 dimensional vector. It's main advantage is that is targeted for high speed computational encoding.

3.3.4 Sentence Transformers

Sentence Transformers are offered through the hugging face library [15] and popular transormers implementations (BERT, RoBERTa, XLM-RoBERTa, & Co.) which are tuned and fit for semantic search problems. The transfer learning model is selected based on the benchmarks presented in [16] where different transformer architectures where evaluated on STSb dataset. Tests were performed using a BERT-large model trained on SNLI and STSb data. 1024 dimension sentence vectors are produced from the mean of the token representation.

Transformer models use WordPiece or SentencePiece tokenizer which practically alleviates the OOV problem which sets a serious drawback in Word2Vec model.

3.4 K-most relevant queries

We estimate the similarity between embeddings using the cosine-similarity matrix

$$\cos\theta_{i,j} = \frac{u_i^T u_j}{\|u_i\|\|u_j\|} \tag{3.1}$$

where u_i,u_j are the given vector representations.

There are three approaches from we can extract the k-most similar:

1. **Sentence wise comparison between set elements**: To extract the similarity score between two queries, we compare all question examples from the one with all of the other. For example: If Q1 set has 3 questions in it and Q2 has 2, we get 6 similarity scores in total, from which we keep the maximum representing the score between Q1,Q2.

2. **Question Centroid comparison**: Each query set is represented as a centroid that is calculated by performing a pooling operation on the within set question embeddings.

3. **QA Centroid comparison**: Similarly to the previous approach, the query centroid is now a centroid from pooled vector that contains the question/answer centroid.

Since the maximum group size is 5 the default K configuration was set to 5.

At this point we have to emphasize that the similarity score is estimated in two ways. The first, takes into account a **safe rule** which assigns a similarity score of 1 in query pairs that share the same answer. The second case estimates the similarity strictly based on the query's vector representation.

Another valid option is to select the most similar queries given a universal threshold. One possible drawback this might have, is that is possible bringing false positives for a query and cut off relevant queries for another. This claim is further analyzed in Section 4.2.

Chapter 4

Experiments

4.1 Evaluation Metrics

4.1.1 Mean Reciprocal Rank (MRR)

The mean reciprocal rank [17] is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

where, $rank_i$ refers to the rank position of the first relevant document for the i-th query, and Q is a sample of queries.

The MRR generaly applies in cases where there is one correct response. Since we have multiple correct responses we estimate the rank when the first estimated suggestion is in the group

4.1.2 Recall@K

$Recall@K$ (Manning et al. [18]) is the percentage of relevant items selected out of all the relevant items in the repository at the K recommended items.

$$Recall@K = \frac{S_t(K)}{R_t}$$

where,

K is the number of queries to be recommended,

S_t is the number of the recommended items that are relevant in the top K suggestions,

R_t is the total relevant items for each query.

$Recall@K$ penalizes strongly when the recommended relevant items are less than than the actual relevant.

4.1.3 R-Precision@K

For a given query topic Q, $R-precision$ is the precision at $\min(K, R)$, where R is the number of relevant items for Q and K are the recommended items . In other words, if there are r relevant documents among the top-R retrieved items, then $R-precision$ is $r/\min(K, R)$. This metric is a modification of the $R-precision$ metric stated in [18]. It was proposed by Chalkidis et al. [19], and is suggested to lead in a more informative and fair evaluation.

4.1.4 Normalised Discounted Cumulative Gain (nDCG@K)

$DCG@K$ (Manning et al. [18]) for K shown recommendations sums the relevance of the shown items for the current query (cumulative), meanwhile adding a penalty for relevant items placed on lower positions (discounted).

$$DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log(i+1)}$$

The Normalized Cumulative Gain for K shown recommendations ($nDCG@K$) divides this score by the maximum possible value of $DCG@K$ for the current query i.e. what the score $DCG@K$ would be if the items in the ranking were sorted by the true (unknown for the recommender model) relevance. This is called Ideal Discounted Cumulative Gain ($IDCG$).

$$nDCG_K = \frac{DCG_K}{iDCG_K}$$

4.2 Results

On the initial dataset, applying the proposed methodology we obtained the following results:

From Table 4.1 we can deduct that both USE and SBERT surpass the baseline models which is an indicator the we are able to detect semanticaly similar queries. This claim is validated further by the difference in results using the safe rule where in the BoW models the differences are greater than the rest. Also not taking into account the safe rule leads in reduced performance. This is more evident in cases where the answer set has not been considered.

Embed Method	**Metrics**			
	MRR	Recall@5	R-precision@5	nDCG@5
Word2Vec: Q Centroid	0.324 | 0.503	0.482 | 0.625	0.482 | 0.625	0.537 | 0.645
TFIDF	0.482 | 0.661	0.548 | 0.685	0.548 | 0.685	0.582 | 0.696
TFIDF Weighted Word2Vec: Q Centroid	0.462 | 0.605	0.557 | 0.738	0.557 | 0.738	0.562 | 0.746
USE Pairwise	0.551 | 0.69	0.661 | 0.786	0.661 | 0.786	0.623 | 0.726
USE Q Centroid	0.509 | 0.613	0.661 | 0.786	0.661 | 0.786	0.589 | 0.669
USE QA Centroid	0.759 | 0.759	0.729 | 0.738	0.729 | 0.738	0.751 | 0.751
SBERT Q centroid	0.580 | 0.643	0.622 | 0.622	0.622 | 0.622	0.641 | 0.691
SBERT QA Centroid	0.758 | 0.758	**0.789** | **0.789**	**0.789** | **0.789**	**0.782** | **0.782**

Table 4.1: Benchmarks on Helvia Dataset. Each metric has two values; the first | second indicating that the safe rule is deactivated/activated. For each embedding method, the best models are selected after fine tuning on the pooling operation in order to obtain the best parameter combination.

In order to assess the difficulty of the problem we are trying to solve, some descriptive statistics are presented in the form of boxplots:

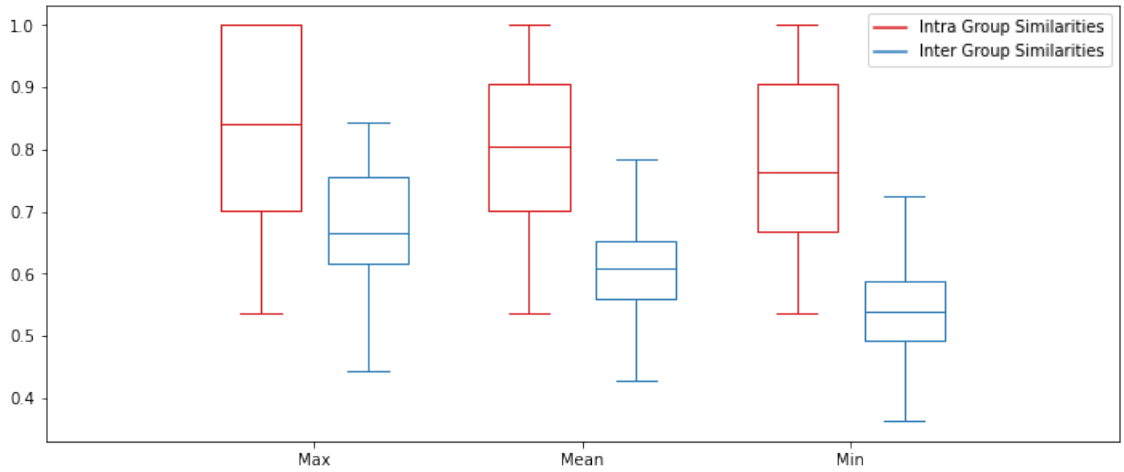

Figure 4.1: Boxplot Comparison between Intra/Inter group similarities.From this graph we observe, that the average maximum inter group similarity is greater than the minimum max of intra group similarity. Setting a threshold would result fetching in some cases too many false positives which pose an overhead for the curator to filter similar intents.

K Selection

Since this system is oriented for business intelligence purposes, it's wise to assume that the curator will have to process the data with speed and as less verbose information possible. On that basis, we have to search for the optimal K values for filtering the queries. In Figure 4.2 are shown the recall curves which is a better suited metric for this task.

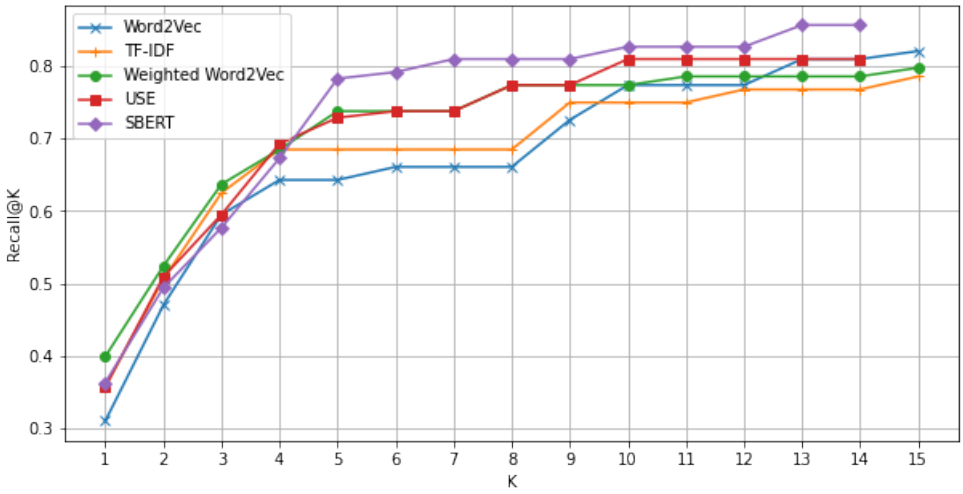

Figure 4.2: Recall values for different K

We can see, for $K > 10$ the models start to get saturated with irrelevant recommendations. The baseline models converge slower to the optimal solution than USE and SBERT.

4.3 Human in the Loop Evaluation

4.3.1 Initial Approach or a semi-automated grouping process.

The main goal is to evaluate the processing time needed for a human to filter similar groups, with and without the assist of a tool that detects semantically similar intents. This task is assigned to Helvia's data curators, where their role is to refine the initial data in order to improve the overall Client-NLU system experience. To achieve that, we compare the time that took 2 data curators to report their findings, given an upper time limit. Also, we want to compare how many similar pairs have been detected in both occasions. This assessment is applied on a new dataset consisting of 362 intents.

The tool offers two functionalities:

1. Display the maximum global similarities between intents. Essentially, we fetch the most similar intent for a given query Q in descending order for the top K most similar. This serves as a filtering function to look for the most profound similarities, and in a follow-up step checks further for existing similarities.

2. Given a specific query, the K most relevant ones suggested. This results in detecting multiple similar pairs thus creating in a final step groups of similar intents. For this assessment, Helvia Technologies used only the first utility of the tool.

In addition, the data curators have no prior knowledge of the new dataset. For that purpose they are given 15 minutes to familiarize themselves with it. Then we begin the assessment process as described below:

First, the task is to find similar intents for a given period of 20 minutes, which is repeated for another 20 (40 minutes in total), without the aid of the tool. In addition, they provide a level of a certainty (High, Medium, Low) for each found pairing. The same time limit applies to the evaluation of the assisting service. In detail, the curators are asked to evaluate the top 25 most similar pairings suggested by the 2 embedding encoders based on Sentence Bert. The first, takes into account only the Question set while the second takes into account both Question/Answer sets. Both are selected according to the performance on the initial dataset presented in Section 3.1.

	Results	
	Time (minutes)	Detected Similarities
Curator 1	40 (20+20)	16 (10+6)
Curator 2	40 (20+20)	15 (11+4)
Curator 1 + Model 1	10	13
Curator 1 + Model 2	10	11
Curator 2 + Model 1	10	15
Curator 2 + Model 2	10	11

Table 4.2: Assessment of the assisted matching of similar sets against manual evaluation. The first column suggests the origin of the results. In the second column, it's displayed the total time that the curators have taken to report their findings. In the third column, the curator's detected similarities are reported, where the (a+b) notation refers at the reported findings for each time period. With Model 1 and 2 are suggested the SBERT models that consider Q and QA centroid representations respectively.

From Table 4.2, it is evident that using the aided matching system the time needed for finding similar sets is substantially lower than the manual evaluation, while the reported similarities are not significantly different.

In detail, in the first two rows, where the outcome of the manual evaluation is reported, both curators used all the time that had at their disposal (40 mins). On the other hand, with the assist of the tool, which targeted specific intents, the process took only about 10 minutes for both curators. Examining the "Detected Similarities" column, the reported findings are very close to that of the manual evaluation. Specifically, for Curator 1 during the manual assessment, he reported in total 16 similar intents where 10 were reported in the first 20 minutes and 6 in the second 20 minute period. Using the assisting tool, which considered only each intent's Questions phrases, he reported 13 similar intents in 10 minutes which is marginally higher than his manual report during the first period.

An interestingly fact is that the SBERT model which takes into account only the Questions set detects more pairings than the model which considers both QA sets.

At this point we have to mention that the comparison is slightly unfair. This is due to the fact that manual evaluation is performed in an all-vs-all manner, meaning, comparing each query with the rest of the dataset. On the other hand, we evaluate only the top 25 recommendations. A more solid approach is to include the second utility of the tool as referred above. Another reason is that, there should be 4 instead of 2 curators where, the first two would perform the manual assessment and the other 2 would use the assisting service.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Our experimental results showed that using pretrained models on semantic similarity tasks models, improves the detection of similar queries. This becomes more apparent in cases where semantically related queries are matched whereas the baseline models are not able to detect. Additionally, in our case, where data are insufficient for training in order to obtain a specific domain knowledge for our sentence embbedings, models such as USE and SBERT proved to be useful.

Finally, through the Human in the Loop evaluation, it becomes clear that an automated assisting tool for this task is approximately 4 times faster than the manual processing time.

5.2 Future Work

Interactive Clustering

Real-world data may include various possible pairings, and a fully unsupervised clustering has no means of creating a grouping that fit the user's requirements, this is due to that external domain knowledge is needed. Also, quality of a clustering outcome relies heavily on inferring the proper features as well as specifying appropriate similarity measures. In addition, several parameters are typically required for example, the number of intended clusters or the minimum cluster size. Given these requirements, a real-world clustering task can be too complex to be solved fully automatically.

For the aforementioned reasons, there is a need for solutions that require end-users to interact directly with the clustering process to tune it to certain application domains and permit it to systematically adapt to their preferences. Several inter-

active clustering techniques have lately proposed in which the user and the system interact with each other to carry out the clustering task. In [20] [21] this type of clustering framework is explored which is fitted as well for this project's specifications. In detail, The framework models are based on the principal of posing practical constraints on the clustering algorithms. Starting with an initial clustering that it cannot be modified arbitrarily,it is only allowed to make local edits consistent with user requests.On this premise, the researchers develop several simple, yet effective algorithms under different assumptions about the nature of the edit requests and the structure of the data. To be considered this frameworks useful, it is necessary the algorithms should converge to the target clustering after a small number of edit requests.

Bibliography

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JOURNAL OF MACHINE LEARNING RESEARCH*, 3:1137–1155, 2003.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[4] A. Kosmopoulos, Ion Androutsopoulos, and G. Paliouras. Biomedical semantic indexing using dense word vectors in bioasq. 2015.

[5] Craig W. Schmidt. Improving a tf-idf weighted document vector embedding. *CoRR*, abs/1902.09875, 2019.

[6] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015.

[7] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[8] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[9] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017.

[10] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics.

[13] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675, 2019.

[14] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[15] https://huggingface.co/.

[16] https://docs.google.com/spreadsheets/d/14QplCdTCDwEmTqrn1LH4yrbKvdogK4oQvYO1K1aPR5M/edit#gid=0.

[17] https://web.stanford.edu/class/cs276/handouts/evaluationnew-handout-1-per.pdf.

[18] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[19] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. Large-scale multi-label text classification on EU legislation. *CoRR*, abs/1906.02192, 2019.

[20] Maria-Florina Balcan and A. Blum. Clustering with interactive feedback. In *ALT*, 2008.

[21] Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18(3):1–35, 2017.

Appendices

Experimental Results on different K values

Embed Method	**Metrics**			
	MRR	Recall@1	R-precision@1	nDCG@1
Word2Vec Q Centroid	0.17	0.31	0.607	0.607
TFIDF embeddings	0.35	0.357	0.679	0.679
TFIDF weighted Q Centroid	0.285	0.399	0.714	0.714
USE QA Centroids	0.714	0.357	0.714	0.714
BERT QA Centroids	0.714	0.363	0.714	0.714

Table 1: Evaluation results on the initial dataset (105 QA sets) for K=1

Embed Method	**Metrics**			
	MRR	Recall@10	R-precision@10	nDCG@10
Word2Vec Q Centroid	0.56	0.774	0.774	0.677
TFIDF embeddings	0.704	0.75	0.75	0.715
TFIDF weighted Q Centroid	0.609	0.774	0.774	0.758
USE QA Centroids	0.749	0.81	0.81	0.767
BERT QA Centroids	0.754	0.827	0.827	0.787

Table 2: Evaluation results on the initial dataset (105 QA sets) for K=10

Performance Evaluation Curves on different K values

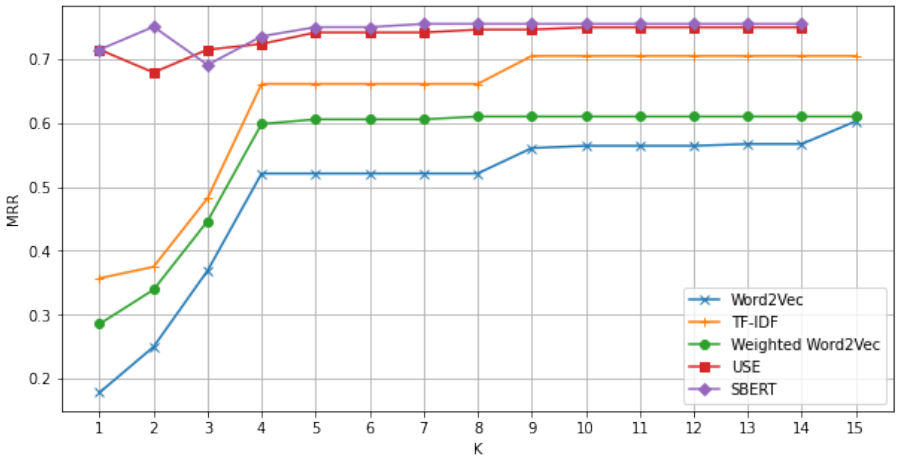

Figure 1: MRR curves for different K

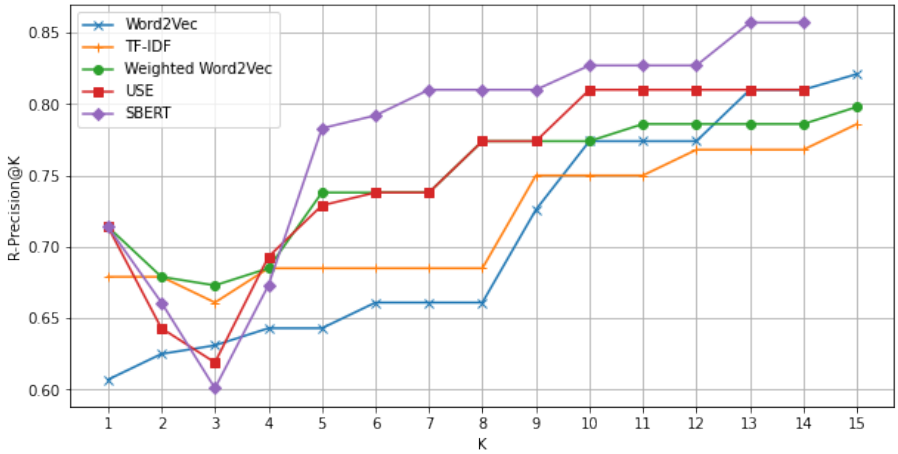

Figure 2: R-Precision curves for different K

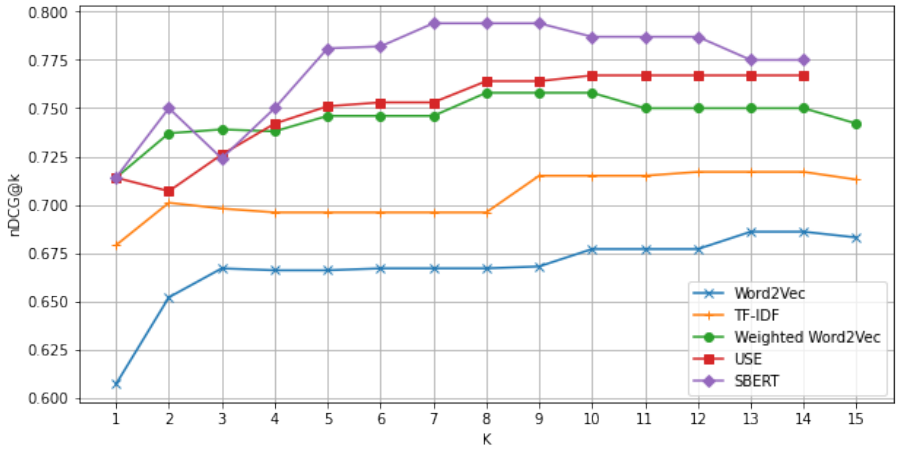

Figure 3: nDCG curves for different K