Design and Development of an Efficient Semantic Knowledge-based Technique for Understanding Short Texts

Sarada Sreekumar[1], Shiji S[2]
[1]*M.Tech. Scholar, Dept. of CSE, NCERC, Kerala*
[2]*Assistant Professor, Dept. of CSE, NCERC, Kerala*

Abstract - The correct interpretation of the meaning of the short texts is a difficult task for machines. The key factor that has to be identified is what concept the short text is conveying. For short texts such as search queries, news titles or tweets, traditional methods ranging from part-of-speech tagging to dependency parsing find it difficult to capture the underlying semantics. In order to better understand the short texts, it is highly essential to exploit semantic knowledge. Semantic knowledge is acquired from a well-known database namely Probase. The three major tasks in this paper include text fragmentation, type discovery and conceptualization, and all these tasks exploit the semantic knowledge. Further, our paper takes into account the spatial-temporal factors. The experimental results show that our system is efficient in understanding short texts.

Keywords: short text, text fragmentation, type discovery, conceptualization, semantic knowledge.

I. INTRODUCTION

Most of the applications and social networking sites deal with short texts. Short texts refer to texts with limited content (two to eight words). It consists of only a small number of keywords. The data mining and information retrieval techniques currently deal with a huge volume of short texts and it has become highly essential to analyze what the short text is conveying. The reasons for the difficulty in understanding short texts include: (i) Short texts usually do not follow the syntax of the written language; (ii) Short texts do not contain sufficient statistical signals to support the recent developments in text mining such as topic modeling; (iii) Short texts are ambiguous and noisy and are generated in large volumes; (iv) The scarcity of contextual information further makes it complex for the machines to understand them.

The most important task in understanding short text is to identify the semantics hidden in it. Many studies have been conducted in this field [1] [2] but most of the techniques do not take into account the contextual information. It is very easy for humans to interpret the meaning of short texts but, in the case of machines, it is a hazardous task. According to the well known psychologist, Gregory Murphy, "concepts are the glue that holds our mental world together" [3]. So the key aspect of understanding short text is to determine the concepts outlined in the short text.

The major challenges faced by short texts include:

- Incorrect fragmentation

  Given a short text, the way in which it has to be fragmented is a challenging task. A short text can have multiple possible fragmentations. For example, 'hotel california eagles' vs. 'stay hotel california'. 'hotel california' can be fragmented as '<u>hotel</u> <u>california</u>' or '<u>hotel california</u>'. So, the semantic coherence need to be considered while determining the correct fragmentation. The most frequently used Longest Cover method [4] [5] [6] [7] [8] determines the longest fragment as the best fragment which can result in 'stay hotel california' being fragmented as '<u>stay</u> <u>hotel california</u>' which is an incorrect segmentation as '<u>hotel california</u>' as a single fragment is an instance of concept song.

- Abbreviations and noisy short texts

  A single abbreviation can have multiple expansions. Only the correct semantic interpretation of the short text can result in the right expansion at the right place. Another issue with short texts includes employing nicknames. The vocabulary should be designed in such a way that it includes as much information about abbreviations and nicknames as possible.

- Multiple type possibilities

  A single word can have numerous types. For example, 'sharp products' vs. 'sharp knives'. The different possible types of sharp include instance and adjective. So, determining the best type based on semantics is another major challenging task.

- Ambiguous instance

An instance can belong to multiple concepts and the concept to which it belongs must be determined based on semantics. For example, 'read Sherlock Holmes' vs. 'watch Sherlock Holmes' vs. 'age Sherlock Holmes'. In the first case, Sherlock Holmes is an instance of concept book whereas an instance of concept movie in the second case and character in the third case. So, the concept to which an instance belongs can be determined only if proper interpretation of the semantics of the short text is being performed.

- Huge volume of data

Short texts such as tweets, news titles, statuses, comments etc. are generated in such a large volume that we cannot consider the probability of neglecting it. It has been found that in Twitter, an average of 340 million tweets is posted per day and in Google, almost 3 billion queries are placed every day. So, handling short texts is a highly important task.

In this paper, a novel approach is proposed that aims to enable the proper understanding of short texts based on the underlying semantics. Semantic coherence is estimated in all the steps to provide an accurate understanding of the short texts. The experimental results show that the proposed system is highly accurate in creating the required predictions and the best concept of the instances has been identified and labeling is performed.

II. SYSTEM MODEL

The semantic interpretation of the short text is the main objective of our system. The task of understanding short text can be divided into three subtasks which include:

- Text fragmentation: The process of dividing the short text into appropriate fragments such that the fragments are semantically coherent.
- Type discovery: The process of determining the best type for each fragment.
- Concept labeling: For each instance, finding the most appropriate concept cluster.
    - Fig. 2 illustrates our system for short text understanding. It mainly consists of two parts namely static part and the dynamic part. Static part deals with gathering data that is required to process the short text whereas the dynamic part includes the steps that processes the short text and provides the necessary interpretation.

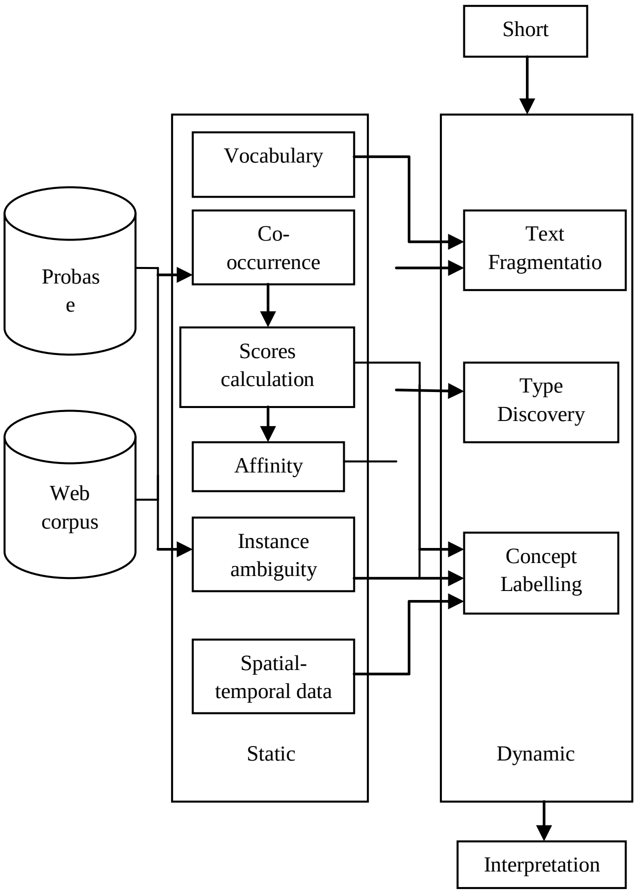

Fig. 2.1 System overview

III. PREVIOUS WORK

In this section, some of the existing techniques are being discussed and the methodologies used are described in brief. All the techniques discussed here provides a comparatively low performance as compared to our system.

3.1 TAGME: On-The-Fly Annotation Of Short Text Fragments

Given a plain text, it is augmented with hyperlinks to Wikipedia pages [6]. Texts which are short and poorly composed are annotated with hyperlinks to Wikipedia. Important terms are identified from the short text and are annotated with unambiguous entities (usually Wikipedia pages) drawn from a catalogue. These terms are called spots and they are treated as Wikipedia anchor texts and the pages linked to them in Wikipedia are possible senses.

3.2 Short Text Conceptualization Using A Probabilistic Knowledgebase

A knowledgebase is a collection of entity, facts, relationships that conforms to a certain data model. A knowledgebase helps machines understand humans, languages and the world. A probabilistic knowledgebase helps the machine to understand the probability that an instance belong to a concept. The mechanism that is used in [7] to conceptualize words and successively the short text is the Bayesian inference mechanism. The probability that an instance belongs to a concept is calculated as follows:

$$p(c|I) = \frac{p(I|c).p(c)}{p(I)} \tag{1}$$

Where I is the instance, c is the concept. In eq.1, p(c|I) is the popularity score (the probability that people think of concept c when seeing the instance I), p(I|c) is the typicality score (the probability that people think of instance I when seeing the concept c), p(c) is the probability of occurrence of concept c in the knowledgebase and p(I) is the probability of occurrence of the instance I in the knowledgebase.

3.3 Context Dependent Conceptualization

The important aspect that has to be considered while understanding short texts is to analyze the context in which it occurs. Conceptualization is the process of mapping a short text to a set of concepts. A probabilistic topic model namely Latent Dirichlet Allocation (LDA) [8] is used to capture the semantic relations between words. The probabilistic topic model along with Probase (knowledgebase) determines the concepts in the short text based on the context.

3.4 Semantic Enrichment (SE) Technique

It mainly consists of four steps namely pre-processing short text, conceptualization, finding co-occurring terms and final concept detection [9].

In the initial step, the short text is broken down into appropriate segments. The longest matching term is found out. The semantic coherence of that term with the rest of the terms in the short text is determined. If there is semantic relationship, that term is considered. Otherwise, the next longest matching term is considered. This process is repeated until the best segmentation is obtained.

The second step is to obtain the set of concepts for each segment term. Here, LDA model is being used. The third step determines all the terms co-occurring with the segments in the short text. Two types of scores are being taken into consideration: co-occurrence probability and semantic similarity.

The final step is to find out the actual concept cluster to which the segments belong to. The co-occurrence probability of each concept cluster with the concept clusters of other terms is considered. The pairs with co-occurrence probability, greater than 0.7 are returned. Then the co-occurrence probability of each selected pair is checked with that of the common co-occurring terms' concept clusters. The pair with the highest co-occurrence probability is considered.

3.5 Harvesting And Analyzing Semantic Knowledge (HASK) Technique

It mainly consists of two parts: offline part and online part [10]. The main steps in offline part include constructing co-occurrence network, affinity score calculation and determination of instance ambiguity, and that of online part include text segmentation, type detection and concept labeling.

The co-occurrence network is constructed to model semantic relatedness. The nodes in the co-occurrence network are typed-terms and edge weight denotes the semantic relatedness between typed-terms. Initially, the nodes in the co-occurrence network are verbs, attributes, adjectives, concepts and instances. That co-occurrence network is compressed in such a way that instances are placed in their appropriate concept clusters. Affinity score is used to measure the semantic coherence between typed-terms. Two typed-terms are coherent if they are semantically similar or they often co-occur in the web. The final step in the offline part is to determine the instance ambiguity level. There are basically three levels of ambiguity which include ambiguity level 0 (Instances that most people consider as unambiguous), ambiguity level 1 (Instances that can be ambiguous or unambiguous) and ambiguity level 2 (Instances that are ambiguous).

Text segmentation is the process of dividing the text into appropriate segments. Here all the possible segments are considered and a term graph is constructed. The best segmentation is obtained using the Monte Carlo maximal clique algorithm. After the best segments are obtained, the type of the segments is detected using pairwise model. Finally, the concepts of the instances in the short text are determined using weighted vote approach.

IV. PROPOSED METHODOLOGY

In this section, the details of our system for short text understanding are described.

4.1 Static Part

The static part includes the following steps: vocabulary construction, scores calculation, affinity score calculation, instance ambiguity determination and spatial-temporal data collection.

4.1.1 Vocabulary construction

The list of English verbs and adjectives are downloaded from an online dictionary namely YourDictionary. The list of attributes, concepts and instances are derived from the knowledgebase, Probase. Verbs, adjectives, attributes, instances and concepts altogether constitute our vocabulary.

4.1.2 Co-occurrence network construction

Co-occurrence network is constructed in order to determine the semantic relatedness between two typed-terms. Typed-terms are terms with a specified type. The nodes in the co-occurrence network are verbs, adjectives, attributes, instances and concepts and the edge weight denotes the semantic relatedness. The co-occurrence network is constructed based on the following assumptions:

- The higher the frequency of two typed-terms co-occurring in a sentence, higher the semantic relatedness.
- The closer the two typed-terms co-occur in a sentence, higher the semantic relatedness.

Based on these assumptions, co-occurrence network is constructed as follows:

- A web corpus is chosen and all the sentences in it are scanned and tagged the words in it as verbs, adjectives and nouns using Stanford POS tagger. For words tagged as nouns, we check whether it is concept, instance or attribute by comparing it with the vocabulary.
- Once the nodes are created, an edge is added between all the nodes and the edge weight is calculated as follows:

$$w(\bar{x},\bar{y}) = \frac{f(\bar{x},\bar{y})}{\sum_{\bar{z}} f(\bar{x},\bar{z})} . \log \frac{N}{N_{nei(\bar{y})}} \quad (2)$$

$$f(\bar{x},\bar{y}) = \sum_{s} f_s(\bar{x},\bar{y}) \quad (3)$$

$$f_s(\bar{x},\bar{y}) = n_s e^{-dist(\bar{x},\bar{y})} \quad (4)$$

In eq. 2, N is the total number of nodes (typed-terms) in the co-occurrence network, $N_{nei}(\bar{y})$ is the number of co-occurrence neighbors of $\bar{y}$. In eq. 3, $f(\bar{x},\bar{y})$ denotes the frequency of two typed-terms appearing together and $f_s(\bar{x},\bar{y})$ denotes the frequency of two typed-terms appearing together in the sentence s. In eq. 4, n_s denote number of times the sentence s appears in the web corpus.

4.1.3 Scores calculation

In this section, we calculate similarity score, co-occurrence score, popularity score and typicality score. The similarity between two typed-terms $\bar{x}$ and $\bar{y}$ is calculated as follows:

$$S_{sim}(\bar{x},\bar{y}) = \cos ine(\bar{x}.\vec{C}, \bar{y}.\vec{C}) \quad (5)$$

Where $\bar{x}.\vec{C}$ is the concept cluster vector of $\bar{x}$ and $\bar{y}.\vec{C}$ is the concept cluster vector of $\bar{y}$. The co-occurrence between two typed-terms is calculated as follows:

$$S_{co}(\bar{x},\bar{y}) = \cos ine(\vec{C}_{co(\bar{x})}, \bar{y}.\vec{C})$$

(6) Where $\vec{C}_{co(\bar{x})}$ denotes the concept cluster vector of the typed-term $\bar{x}$ and it can be retrieved directly from the co-occurrence network. The concept cluster vector is defined as follows:

$$\bar{t}.\vec{C} = \begin{cases} \phi & \bar{t}.r \in \{v, adj, attr\} \\ (<C,1>|\bar{t} \in C) & \bar{t}.r = c \\ (<C_i, W_i>|i = 1,2,.....N) & \bar{t}.r = I \end{cases} \quad (7)$$

Where $\bar{t}$ is the typed-term, $\bar{t}$ .r refers to a specific type, c denotes concept and I denotes instance.

Popularity score is used to determine how often people think of a concept when seeing an instance and typicality score is used to determine how often the people think of an instance when seeing a concept. Popularity score is calculated as follows:

$$p(c|I) = \frac{n(c,I)}{n(I)} \quad (8)$$

Typicality score is calculated as follows:

$$p(I|c) = \frac{n(c,I)}{n(c)} \quad (9)$$

In eq. 8 and 9, n(c,I) denotes the number of times the instance I belongs to the concept c in the web corpus. In eq. 8, n(I) denotes the number of times instance I occur in the web corpus. In eq. 9, n(c) denotes the number of times concept c occurs in the web corpus.

4.1.4 Affinity score calculation

Affinity score measures the semantic coherence between two typed-terms. There are basically two types of coherence which include similarity and relatedness. Eq. 5 in section 4.1.3 denotes the equation for similarity and eq. 6 denotes the equation for relatedness. Affinity score is the maximum of these two scores and it is calculated as follows:

$$S(\bar{x},\bar{y}) = \max(S_{sim}(\bar{x},\bar{y}), S_{co}(\bar{x},\bar{y})) \quad (10)$$

4.1.5 Instance ambiguity level determination

There are basically three levels of ambiguity which include:

- Ambiguity level 0

  Instances that are not ambiguous. Example: Dog (animal)

- Ambiguity level 1

  Instances that can be ambiguous or unambiguous. Example: Google (company and search engine)

- Ambiguity level 2

  Instances that are ambiguous. Example: Puma (company and animal)

$$\text{level} = \begin{cases} 0 & \text{if number of concept clusters is } 1 \\ 1 & \text{if number of concept clusters} > 1 \text{ and after clustering} = 1 \\ 2 & \text{if number of concept clusters} > 1 \text{ and after clustering} > 1 \end{cases} \quad (11)$$

Similar concept clusters are merged together. Two concept clusters can be considered similar if they share a number of common instances. After merging, the level of ambiguity is determined as in eq. 11.

4.1.6 Spatial-temporal data collection

Spatial data is the data or information about the geographic location whereas temporal data is the data involving time. The ambiguity level of the instance is affected by spatial-temporal data. An example of temporal data is Puma Company was founded in 1948 and it is an ambiguous instance only after 1948. Till 1948, it was treated only as animal. Therefore, if a period before 1948 is mentioned in the short text containing Puma, then it should be considered to belong to ambiguity level 0. An example of spatial data is that River bank is an ambiguous instance only in California. It is a regional bank of California. Only in those comments with a relation to California, River bank needs to be considered to belong to ambiguity level 2. Otherwise, it can be considered to belong to ambiguity level 0. All the instances that are affected by these spatial-temporal factors are collected in this module.

4.2 Dynamic Part

Dynamic part includes the following steps: text fragmentation, type discovery and concept labelling.

4.2.1 Text fragmentation

Text fragmentation is the process of dividing the short text into a set of fragments in such a way that:

- Except stop words, each word belongs to one and only one fragment.
- There exists logical consistency among terms.

Initially, we determine all the possible terms from the short text by comparing it with the vocabulary. Each term is considered as a candidate term. We construct a Term Graph (TG) with the candidate terms as nodes and the affinity score calculated in section 4.1.4 as the edge weight. If the affinity score is 0, then we assign an edge weight of 0.001 to apply Monte Carlo algorithm. Apart from this, each node is assigned a weight to denote the coverage of words by that term in the short text excluding stop words. There exist edges only between those terms that are mutually exclusive (these terms does not share any word in common, they form disjoint terms). For example, in the short text, ‘april in paris lyrics’, even though ‘april’ and ‘april in paris’ are candidate terms, there does not exist edges between the terms. Fig 4.1 illustrates the term graph of the short texts ‘april in paris lyrics’ and ‘vacation april in paris’.

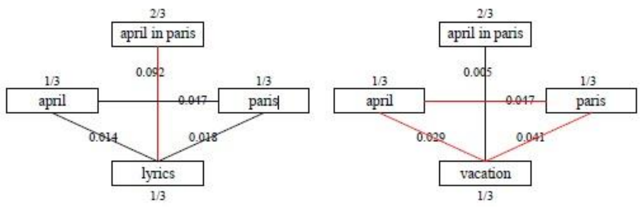

Fig 4.1 Coherent segmentations of ‘april in paris’ and ‘vacation april in paris’ (Wen Hua et al., 2016, p. 8)

Algorithm 1 finds the maximal clique and Algorithm 2 determines how many times the Algorithm 1 must run in order to find the best fragmentation. The best fragmentation can be obtained by applying Monte Carlo Algorithm that finds the maximal clique. The Algorithm 1 returns the combination of edges with maximum average weight and the set of nodes that are interconnected by these edges form the best segmentation.

Algorithm 1 Maximum Clique by Monte Carlo

```
Input:
    G = (V, E); W(E) = {w(e)|e ∈ E}
Output:
    G’ = (V’, E’); s(G’)
1: V’ = $\phi$ ; E’ = $\phi$
2: while E ≠ $\phi$ do
3:     randomly select e = (u, v) from E with probability proportional to its weight
4:     V’ = V ∪ {u,v}; E’ = E ∪ {e}
5:     V = V - {u,v}; E = E – {e}
6:     for each t ∈ V do
7:         if e’ = (u,t) ∉ E or e’ = (v,t) ∉ E then
```

```
8:                  V = V – {t}
9:                  remove edges linked to t from
                    E: E - {e' = (t,*)}
10:         end if
11:     end for
12: end while
13: calculate average edge weight: $s(G') = \frac{\sum_{e \in E'} w(e)}{|E'|}$
```

Algorithm 2 Chunking by Maximal Clique

```
Input:
    G = (V, E); W(E) = {w(e)|e ∈ E}
    Number of times to run Algorithm 1: k
Output:
    G'_best = (V'_best, E'_best); s(G')
1: $s_{max} = 0$
2: for i=1; i ≤ k; i++ do
3:     run Algorithm 1 with $G'_i = (V'_i, E'_i)$; $s(G'_i)$ as output
4:     if $s(G'_i) > s_{max}$ then
5:         $G'_{best} = G'_i$ ; $s_{max} = s(G'_i)$
6:     end if
7: end for
```

4.2.2 Type discovery

Once the best fragmentation is obtained, the next task is to determine the type of fragments. In most cases, the fragments will have different types. For example, consider the short text 'watch free movie'. The typed-terms of watch include {$watch_{[c]}$, $watch_{[I]}$, $watch_{[v]}$} where c stands for concept, I for instance and v for verb. The typed-terms of free include {$free_{[v]}$, $free_{[adj]}$} and that of movie include {$movie_{[c]}$, $movie_{[I]}$}.

The best type should be determined based on the semantic coherence. The pairwise model is used for finding the best type. The pairwise model determines the semantic relations between all the terms in the short text. For example, in 'watch free movie', it finds the semantic relation between {watch, free}, {free, movie} and {watch, movie} taking into consideration all the set of possible types. The semantic relation is calculated as follows:

$$w(\bar{x}, \bar{y}) = S_{sg}(\bar{x}) \cdot S(\bar{x}, \bar{y}) \cdot S_{sg}(\bar{y}) \tag{12}$$

$$S_{sg}(\bar{x}) = \begin{cases} 1+\theta & \bar{x}.r = pos(\bar{x}) \\ 1 & otherwise \end{cases} \tag{13}$$

$S(\bar{x}, \bar{y})$ is the affinity score between typed-terms $\bar{x}$ and $\bar{y}$, $S_{sg}(\bar{x})$ is the singleton score of $\bar{x}$. Finally, it constructs a maximum spanning tree and guarantees that the edges of the maximum spanning tree have the largest weight. Kruskal algorithm is being employed in our model to construct the maximum spanning tree. The nodes connecting these edges are returned and that is considered as the best type.

4.2.3 Concept labelling

Initially, we check whether the short text is affected by spatial-temporal parameters and if yes, the ambiguity level of the instances is re-determined. Then, we consider the maximum spanning tree obtained in the type discovery step. If the instance is of ambiguity level 0, then directly assign the concept cluster since there exists only one concept cluster for that instance. If the instance is of ambiguity level 1 or 2, there are basically four different cases that need to be addressed.

Case 1: If verb, attribute or adjective is used to disambiguate an instance, then find the concept cluster of the instance that has maximum co-occurrence value (obtained from the co-occurrence network) with that corresponding verb, attribute or adjective. Whichever yields the maximum value, the instance should be labelled with that concept cluster. Example: 'eating apple'. 'Eating' is the verb and 'apple' is the instance. The concept cluster of apple with highest co-occurrence value with the verb 'eating' is 'fruit'. So, 'apple' is labelled with the concept 'fruit' in this short text.

Case 2: If the instance of ambiguity level 0 is used to disambiguate an instance of level 1 or 2, then directly determine the concept cluster with highest similarity or co-occurrence (based on eqns. 5 and 6 in section 4.1.3). Example: 'dog and puma'. The ambiguity level of 'dog' is 0 and the only concept cluster to which it belongs is 'animal'. The instance 'puma' belongs to ambiguity level 2. Initially consider all the concept clusters of 'puma' which include brand, company, animal etc. The concept cluster which has highest similarity score with the concept cluster of 'dog' is 'animal'. So, the instance 'puma' is labelled with the concept 'animal' in this short text.

Case 3: If the instance of ambiguity level 1 is used to disambiguate an instance of ambiguity level 2, then mutual disambiguation happens. Initially, determine the two different concept clusters of the instance of ambiguity level 2. For example, if the instance is apple, then the two concept clusters which we consider would be 'fruit' and 'company'. Then, label the first concept cluster (fruit in this case) as 0 and the second concept cluster (company in this case) as 1. Then, we derive the sub-concept clusters of 0 and 1 and label them as a and b respectively. We then retrieve the co-occurring concept clusters (of all the concept clusters of the instance) from the co-occurrence network and label them as cn. Now, we have to find the top cluster of the instance of ambiguity level 1 based on the typicality score (eq. 9 in section 4.1.3).

Case 3a: If the top cluster of the instance of ambiguity level 1 belongs to 0 or a, then we neglect the concept clusters under label 1, b and cn. If the top cluster of the instance of ambiguity level 1 belongs to 1 or b, then we neglect the concept clusters under label 0, a and cn. In both the cases, we create a matrix in such a way that we plot the concept clusters (that has not been neglected) of the instance of ambiguity level 2 along rows and that of ambiguity level 1 along columns. Each entry of the matrix will be the average of the popularity score (eq. 8 in section 4.1.4).

Case 3b: If the top cluster of the instance of ambiguity level 1 belongs to cn, then we check to which concept cluster of ambiguity level 2 is that concept cluster co-occurring. Then, we construct the matrix as in case 3a. The second matrix is constructed in such a way that the row will be the instance of ambiguity level 2 and the columns correspond to the concept clusters of ambiguity level 1. Each entry in the matrix will be the co-occurrence value of that instance with each concept cluster (obtained from co-occurrence network). Similarly, we construct third matrix in such a way that the row value corresponds to the instance of ambiguity level 1 and the column value corresponds to the concept clusters of ambiguity level 2. Then, the entries in the first column of the first matrix are multiplied with the first entry in the second matrix; the entries in the second column of the first matrix are multiplied with the second entry in the second matrix and so on. Similarly, the entries in the first row of the first matrix are multiplied with the first entry in the third matrix; the entries in the second row of the first matrix are multiplied with the second entry in the third matrix and so on. The first matrix is altered with the average of these two products. Whichever entry has the maximum value, the corresponding concept cluster pair is returned.

Case 4: If the instance of ambiguity level 2 is used to disambiguate an instance of ambiguity level 2, then we initially determine the two main concept clusters and create a similarity/co-occurrence matrix. The concept cluster pair with the maximum similarity/co-occurrence value is returned. Then we proceed as in case 3.

V. EXPERIMENTAL RESULTS

The two techniques, HASK technique and the proposed technique are executed and tested for various inputs. It was tested with an input set of 100 short texts taken from various social media and online forums. The output was compared with the manual interpretation of the short text and based on that the effectiveness of the techniques was measured.

Accuracy is the measure of how correctly the system could predict the short text. It is calculated by determining how much percentage of the outputs is correct. The output is compared with the manual interpretation of the short text and then accuracy is determined. Fig. 5.1 shows the comparison of HASK and the proposed technique on the basis of accuracy. The techniques are plotted along the x-axis and their corresponding accuracy on the y-axis.

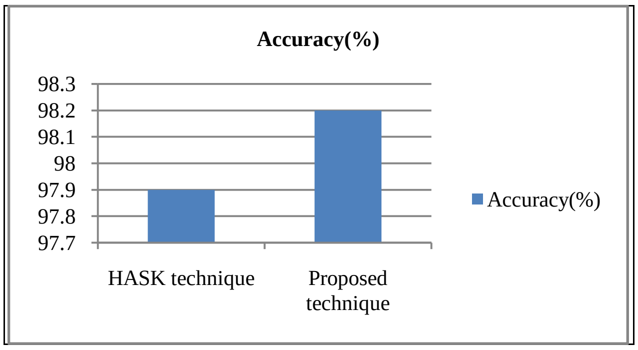

Fig 5.1 Comparison plot of HASK and proposed techniques on the basis of accuracy

When the short text given was ‘eating apple’ (case 1), ‘dog and puma’ (case 2), ‘dog and tiger’ (case 2), ‘vacation april in paris’ (case1), ‘watch free movie’, ‘april in paris lyrics’(case 1), ‘apple and puma’ (case 4) both the techniques responded in the same manner. When the input was ‘tiger and puma’ (case 3a), the output of the HASK technique was ‘tiger CONCEPT animal puma CONCEPT animal’ but the output of the proposed technique was ‘tiger CONCEPT cat species puma CONCEPT cat species’. When the input was ‘apple and ipad’ (case 3b), the output of the HASK technique was ‘apple CONCEPT company ipad CONCEPT device’ but the output of the proposed technique was ‘apple CONCEPT company ipad CONCEPT tablet’. When the short text given was ‘River bank loan California’ (spatial factor case), the output of the HASK technique was ‘River bank CONCEPT moist area loan ATTRIBUTE California CONCEPT state’ but the output of the proposed technique was ‘River bank CONCEPT bank loan ATTRIBUTE California CONCEPT state’. An input supporting temporal data is ‘1947 puma’ (temporal factor case). The HASK technique took more execution time for interpreting this input but our proposed technique took comparatively less time. Table 1 shows the comparison of HASK and the proposed technique on the basis of a few inputs.

TABLE 1. COMPARISON OF HASK AND PROPOSED TECHNIQUES

S. No.	Inputs	HASK technique accuracy	Proposed technique accuracy
1.	Eating apple	100%	100%
2.	Dog and puma	100%	100%
3.	Dog and tiger	100%	100%
4.	Tiger and puma	75%	100%
5.	Apple and ipad	50%	100%
6.	Vacation april in paris	100%	100%
7.	Watch free movie	100%	100%
8.	April in paris lyrics	100%	100%
9.	Apple and puma	100%	100%
10.	River bank loan California	66.7%	100%

VI. CONCLUSION

In this work, an efficient method for short text understanding is proposed. The spatial and temporal factors are considered while interpreting the short text and the concept labeling has been performed in such a way that we considered all the possible cases. The concept labeling algorithm finds the more specific concept of the instance than the generalized approaches. Also, concept labeling solves the problem of assigning the most appropriate cluster to the instances of ambiguity level 1 based on semantic coherence. The experimental results proves that the proposed system provides a more specific and accurate output than the existing technique.

VII. FUTURE SCOPES

As a future work, we attempt to further optimize the concept labeling algorithm. Also, the abbreviations, mis-spelt words and the message languages will be considered. Further, we will try to use SVM classifier for text classification.

REFERENCES

[1] A. McCallum and W. Li, “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons”, in Proceedings of the Seventh Conference on Natural Language Learning, at HLT-NAACL 2003 - Volume 4, ser. CONLL ’03, Stroudsburg, PA, USA, pp. 188–191, 2003.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation”, J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.

[3] G. L. Murphy, “The big book of concepts”, MIT press, 2004.

[4] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee, “Twiner: Named entity recognition in targeted twitter stream”, in Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR ’12, New York, NY, USA, pp. 721-730, 2012.

[5] D. M. de Oliveira, A. H. Laender, A. Veloso, and A. S. da Silva, “Fsner: A lightweight filter-stream approach to named entity recognition on twitter data”, in Proceedings of the 22nd International Conference on World Wide Web, ser. WWW ’13 Companion, Republic and Canton of Geneva, Switzerland, pp. 597–604, 2013.

[6] P. Ferragina and U. Scaiella, “Tagme: On-the-fly annotation of short text fragments (by wikipedia entities)”, in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ser. CIKM ’10, New York, NY, USA, pp. 1625–1628, 2010.

[7] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, “Short text conceptualization using a probabilistic knowledgebase”, in Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three, ser. IJCAI’11, pp. 2330–2336, 2011.

[8] D. Kim, H. Wang, and A. Oh, “Context-dependent conceptualization”, in Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, ser. IJCAI’13, pp. 2654–2661, 2013.

[9] Zheng Yu, Haixun Wang and Min Wang, “Understand short texts through semantic enrichment and hashing”, IEEE Transactions on Knowledge and Data Engineering., vol. 28, no. 2, pp. 566-579, 2016.

[10] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng and Xiaofang Zhou, “Understand short texts by harvesting and analyzing semantic knowledge”, IEEE Transactions on Knowledge and Data Engineering, vol. 29, pp. 499 - 512, no. 99, 2016.