UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Master's Programme in Computer Science

[Title]

[Subtitle]

by

[Colin Manuel Fernandes colin.fernandes@studenti.unimi.it]

F94
Master's Thesis (39 credits ECTS)
(Month) (Year)
Supervisor: [Full name]
Examiner: [Full name]
Word Count:

Abstract Ontology is most efficient and user intuitive way of understanding a subject area or domain. Creating or updating ontologies is a manual, time consuming and expensive. Ontologies also may be written by different people and representation of concepts can be specific and unique to one's style. To update or create ontologies without human intervention, an efficient electronic procedure is needed to identify terms from an existing set or corpus. These newly identified terms would add to the existing terms or concepts or help updating or building a new ontology.

The thesis proposes two methods to generate related terms from initial set of words in a given corpus. Comprehensive overview of wide of range of techniques in text representations using embedding models and also statistical measures for grouping similar text representations are summarised. An algorithm is hypothesised and tested on a corpus replicating the exiting problem. The results of the system are validated against an existing knowledge-base. Finally the findings of the validation are concluded with results on set of terms from the corpus.

You shall know a word by the company it keeps.
- John Rupert Firth 1957:11

1

Introduction

1.1 Problem Statement

An ontology encompasses a representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities that substantiate one, many or all domains of discourse. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject.

A domain ontology (or domain-specific ontology) represents concepts which belong to a part of the world, such as biology or politics. Each domain ontology typically models domain-specific definitions of terms. For example, the word 'Article' has many different meanings. An ontology about the domain of law would model the "Rule" meaning of the word, while an ontology about the domain of "Language" would model as "word" that is used with a noun.

Since domain ontologies are written by people, they represent concepts in very specific and unique ways. As systems that rely on domain ontologies expand, a method needs to be found to update the existing ontology with new terms from the same domain or related terms that were missed to be included.

The problem remains in creating and updating of ontologies. The process is largely manual and therefore time-consuming and expensive. An efficient way needs to be designed to find the new related terms from a corpus to enhance the respective component of the domain ontology.

1.2 Aim and Scope

Given a corpus of documents related to a domain, hypothesise and implement efficient process to find the related terms in the corpus. These terms should aim to help add to the knowledge of existing or new domain ontology.

Given the related terms find a method to validate the relatedness of the algorithm.

1.3 Outline of the Thesis

1. Literature review on current state of the art in automated text processing .
2. Find a dataset to replicate the current problem statement.
3. Hypothesise a method to find new terms and test the algorithms on the dataset.
4. Validation: Design a method to verify the result with existing knowledge base

2

Literature Review

2.1 Text Representations

Representing text as numerical vectors based on relation between target and contexts appear as state of the art for analysing text in a corpus. Also distance between numerical vectors can be found more intuitively than plain text. There are many different ways for learning word embeddings from a corpus ranging from one-hot encoding to dense representations. Intuitively few of them are selected and surveyed to select one for the required task.

2.1.1 A Neural Probabilistic Language Model

The model focuses on learning a statistical model of distribution of word sequences. The model has the vector representations of each word and parameters of the probability function in its parameter set. The objective of the model is to find the parameters that minimize the perplexity of the training dataset. The model eventually learns the distributed representations of each word and the probability function of a sequence as a function of the distributed representations.In this case, even a never-before-seen sentence can obtain a high probability if the words in it are similar to those in a previously observed one.

2.1.2 Word2Vec

Tomas Mikolov et al **?** developed log-linear models which observed large improvements in accuracy and performance at much lower computational cost. In addition using word offset technique using simple vector algebraic operations, it was found that word representations go beyond simple syntactic regularities. For example *vector*(King) - *vector*(Man) + *vector*(Woman) results in a vector that is closest to the vector representation of the word Queen.
The literature proposes two new log-linear models for learning distributed representation of words minimizing the computational complexity.

1. Continuous Bag of words(CBOW): The architecture has 3 layers; an input layer, projection layer and output layer. The input layer is comprised with the sum one hot vector of the words within the context window of the target word. The output layer is one hot representation of the target word. The projection layer weights are trained to map these relation using the words in the corpus.

Finally projection layer will be the vector representation of the output word in a lower the dimension.

2. Continuous Skip Gram Model: The architecture is similar to the CBOW model. In the skip gram the context one hot representation of words are mapped at the output given the one hot representation of the target word.

The computation is still expensive and makes it impractical to scale up to large vocabularies or large training corpora. This problem is solved by limiting the number of output vectors that will be updated for an epoch **?**. This is achieved by hierarchical softmax, negative sampling asynchronous Stochastic Gradient Descent **?**.

2.1.3 Global Vectors (GloVe)

Unlike the word2vec models the Glove model learns word vectors from term co-occurence matrix. A co-occurrence matrix is of size $[V * V]$ where V is the vocabulary size of corpus. Each entry corresponds to the frequency of the word within a window of context. The resulting co-occurrence matrix is transformed to a lower dimension by matrix factorization methods.

Word vectors estimated using GloVe are conceptually similar to those derived from word2vec but uses an underlying count-based model, rather than word2vec's prediction-based model. Because GloVe typically computes statistics over larger context windows than word2vec, it permits capturing longer-term dependencies, although the order of those dependencies will be lost. Empirically, no clear advantage has emerged for either word2vec or GloVe, as the overall performance depends on various factors, including: the type of data and evaluation task being considered.

2.1.4 FastText

Developed by Facebook research is an extension of word2vec with enriched embeddings for subword information. In this approach the embeddings are generated for n-grams using skip-gram model. N - grams are created by moving a window of size 3-6 over the words. These n-grams are used to train a log-linear model with hierarchical softmax. The final word embedding is calculated as the sum of n-gram embeddings that constitute the word. The n-gram information allows sharing the representation across words, thus handling new, out- of- vocabulary terms.

2.1.5 ELMo

ELMo - Embedding from Language Models is a contextualized word- and character-level embedding. Instead of using a fixed embedding for each word, ELMo looks at the entire sentence as it assigns each word an embedding. It uses several stacked bi-directional recurrent neural networks (LSTM) trained on a specific task to create the embeddings. Since it uses a bidirectional architecture, the embedding is based on both the next and previous words in the sentence.

The input to the biLM is computed from characters rather than words, it captures the inner structure of the word. For example, the biLM will be able to figure

Model & Author	**Objective**	**Architecture**	**Notes**
Neural Probabilistic language Model (Bengio Et al . 2003)	Learn Joint probability function of sequence of words in a language, alleviate cure of dimensionality.	Neural Network with Tanh activation and back propagation, coupled with Hierarchical Softmax	First of neural network language model. Not efficient with Scaling.
Word2Vec (Mikolov et al 2013 a,b)	Increasing efficiency of learning embeddings.Given Context window predict target words(CBOW) or vise versa(SG).	Log-linear Model, CBOW/Skip Gram models, coupled with Hierarchical SoftMax.	Word vectors can be somewhat meaningfully combined using just simple vector addition.
Word2Vec (Mikolov et al 2013 c)	Several extensions that improve both the quality of the vectors and the training speed. Subsampling of the frequent words to obtain speed up.	Log-linear Model, CBOW/Skip Gram models, coupled with Negative Sampling.	SGNS (skip-gram with negative sampling), the best performing variant of Word2Vec.
GloVe (Pennington et al)	Improve Semantic similarities with respect to global context, using the insight that co-occurrence ratios, rather than raw counts of.	Log-linear model trained on statistics of word occurrences in a corpus.	Does not cover ambiguity of a words.
MSSG (Neelakantan et al)	Learn embeddings for context and sense jointly	Neural Network: Extension of skip gram model.	Only use local contexts to induce sense representations.
Topical Word Embedding (Liu et al)	Employ topic models with word embeddings. Include complicated correlations among words as well as their contexts 3.	Neural Network Models: Topics obtained by LDA and Gibbs sampling to iteratively assign latent topics.	Fixed number of senses per word.
FastText (Bojanowski Et al)	Model embeddings for out-of-vocabulary words(OOV).	Log-linear model, trained with character n-grams, hierarchical softmax.	Enriching Word Vectors with Subword Information. Use Hashing to improve speed.
ELMo (Peters Et al)	Contextualized word- and character-level embedding.	Neural Network: Multilayer bidirectional LSTM.	Higher level of LSTM States used for Contextualized representations.
BERT (Jacob et al)	Single model to achieve SOTA results in various NLP tasks .	Neural Network: multi-layer bidirectional Transformer pre-trained BERT model.	Fine tune to use the model for various tasks.

Table 2.1: Table of comparison for different embedding models

out that terms like beauty and beautiful are related at some level without even looking at the context they often appear in. ELMo word representations take the entire input sentence into equation for calculating the word embeddings. Hence produced multiple word embeddings per single word for different scenarios. Higher-level layers capture context-dependent aspects of word embeddings while lower-level layers capture model aspects of syntax.

2.1.6 BERT

BERT - Bidirectional Encoder Representations from Transformers is another contextualized word representation model based on a multilayer bi-directional transformer-encoder, where the transformer neural network uses parallel attention layers rather than sequential recurrence. BERT is pre-trained on two unsupervised tasks: (1) a ‘masked language model, where 15% of the tokens are randomly masked (i.e., replaced with the “[MASK]” token), and the model is trained to predict the masked tokens, (2) a ‘next sentence prediction’ (NSP) task, where the model is given a pair of sentences and is trained to identify when the second one follows the first. This second task is meant to capture more long-term or pragmatic information.

2.2 Similarity Measures

After the text is converted into points in vector space, we need to compute similarity between text to find the related terms. Existing work on text similarities are analysed to find an appropriate measure.

Sting based similarity measures are categorised into two, Character based and Term based.

2.2.1 Character-Based Similarity Measures

In Character based measures the comparison is limited to character by character. These measures fail to compare the semantic nature of the words. Words like 'Talk' and 'Speak' are extremely similar semantically but none of the characters are same in both.

1. Longest Common SubString (LCS)
2. Damerau-Levenshtein
3. Jaro
4. Jaro–Winkler
5. Needleman-Wunsch
6. Smith-Waterman
7. N-gram

2.2.2 Term-Based Similarity Measures

This category gives the statistic of distance between words according to the information gained.

Manhattan distance

Also known as Block boxcar distance, absolute value distance, L1 distance and city block distance.It computes the distance that would be traveled to get from one data point to the other if a grid-like path is followed.

Dice's coefficient

Is is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings.

Euclidean distance

L2 distance is the square root of the sum of squared differences between corresponding elements of the two vectors.

Jaccard similarity

Computed as the number of shared terms over the number of all unique terms in both strings.

Word Movers Distance

The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to “travel” to reach the embedded words of another document.

Cosine similarity

Is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

$$\cos(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n}\mathbf{A}_i\mathbf{B}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{A}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{B}_i)^2}} \tag{2.1}$$

2.3 Survey Conclusion

For now BERT and ELMo are complex models, embeddings are good for specific tasks like disambiguation, sentiment analysis, topic modelling or question answering. For now the urge to use these complex models are resisted. In machine learning the simplest solution tends to be comparatively efficient one. And since word vectors are needed only to identify similarities the less complex log - linear models are chosen. FastText becomes an ideal choice as the model handles out of word vocabulary. Word2Vec is also used to test an alternate hypothesis.

For the similarity measure Euclidean and cosine would be a good measure for similarity as they help in measuring the closeness to the said vector. But in an higher dimensional space cosine becomes easier to calculate since its only uses the dot product of vectors in calculation.

3

Library, Packages and Data

3.1 Twitter Dataset

As an experimental database twitter is choose since it is expected to have out-of-vocabulary words (OOV), So the algorithm can be tested for similarities in related words even without exact spellings. The Twitter corpus consists of 400 million twitter microposts (tweets). The database also has less frequent words like url's, numbers etc.

3.2 Pre-Trained Word Embeddings

- Word vectors trained on the above Twitter dataset using FastText are used.
- The approach is beneficial since representations are learned using n-grams.
- Words with similar relatedness will have similar vectors.
- Rare words or words without representation can also be represented with the sum of n-grams.
- Moreover in the twitter data which is subject to spelling mistakes, the original meaning would still be constructed. Eg: 'Macintosh' would have similar representation like ['mcintosh', 'Macintosh.', 'Macintosh!',...]

3.3 Embedding file

FastText embedding file trained by 'Fréderic Godin' are used to test the proposed hypothesis the embeddings are available in the below url path

- `https://drive.google.com/file/d/15zXlbO3bxSYTPt71Fon5-0-oCB8iMSno/view`

3.4 Python Package: Gensim

Gensim is an open-source library authored by Radim Rehurek and developed by RARE Technologies. The package is quite robust and contains methods to load embedding files and object class instances to train and explore the embedding files.

- fasttext - Class to train and explore FastText library
- load_facebook_vectors(binary file) - To load embedding file
- most_similar - get nearest neighbours to a word
- similar_by_vector - get nearest neighbours of a vector
- n_similarity - cosine similarity between list of words

3.5 Knowledge base

3.5.1 WordNet

To validate the hypothesis we use a lexical database of English words. The database has Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts. The synonyms are extracted using python NLTK-wordnet package.

3.5.2 BableNet

Multilingual resource that covers hundreds of languages and, according to need, can be used as either an encyclopedic dictionary, or a semantic network, or a huge knowledge base. Online API are used to query lemma names for each synset ID's.

4

Algorithms

4.1 Algorithm-1

4.1.1 Process Flow

1. Embeddings trained using 'FastText' algorithm on twitter data. Eg. Twitter data and FastText combination is used to test similarities with 'out-of-vocabulary' words. The dimension of keyedvectors is reduced to 100 to fit into the RAM.

2. Nearest neighbours to a query word are taken from using the FastText model. Eg: nearest_neighbours('computer') would give word_list ([laptop, HP computer,. . .])

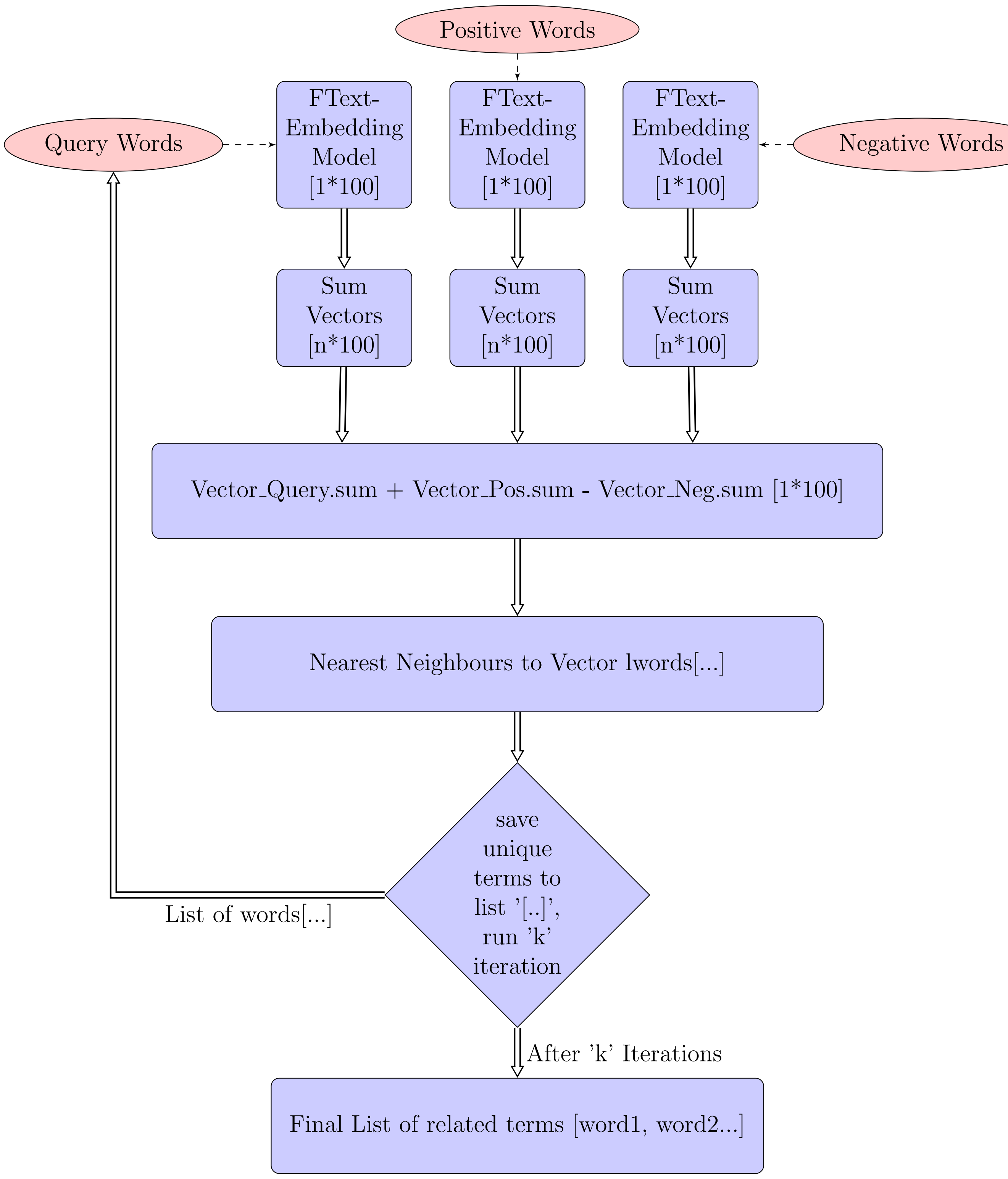
3. The embeddings of the list of words are concatenated together into a matrix. Eg. word_list with 10 words of [1*100] dimension vectors would make [10*100]

4. Mean of the matrix is added with the positive word embedding and subtracted with negative word embedding.

5. 10 nearest neighbor words of the Resulting vector are found and added to a result_list=['phone/computer', 'computer/laptop',..]

6. Step $2- > 3- > 4- > 5$ is considered as 1st epoch and these steps are repeated 'n' times to new words are appended to the result_list which is the final list of related terms.

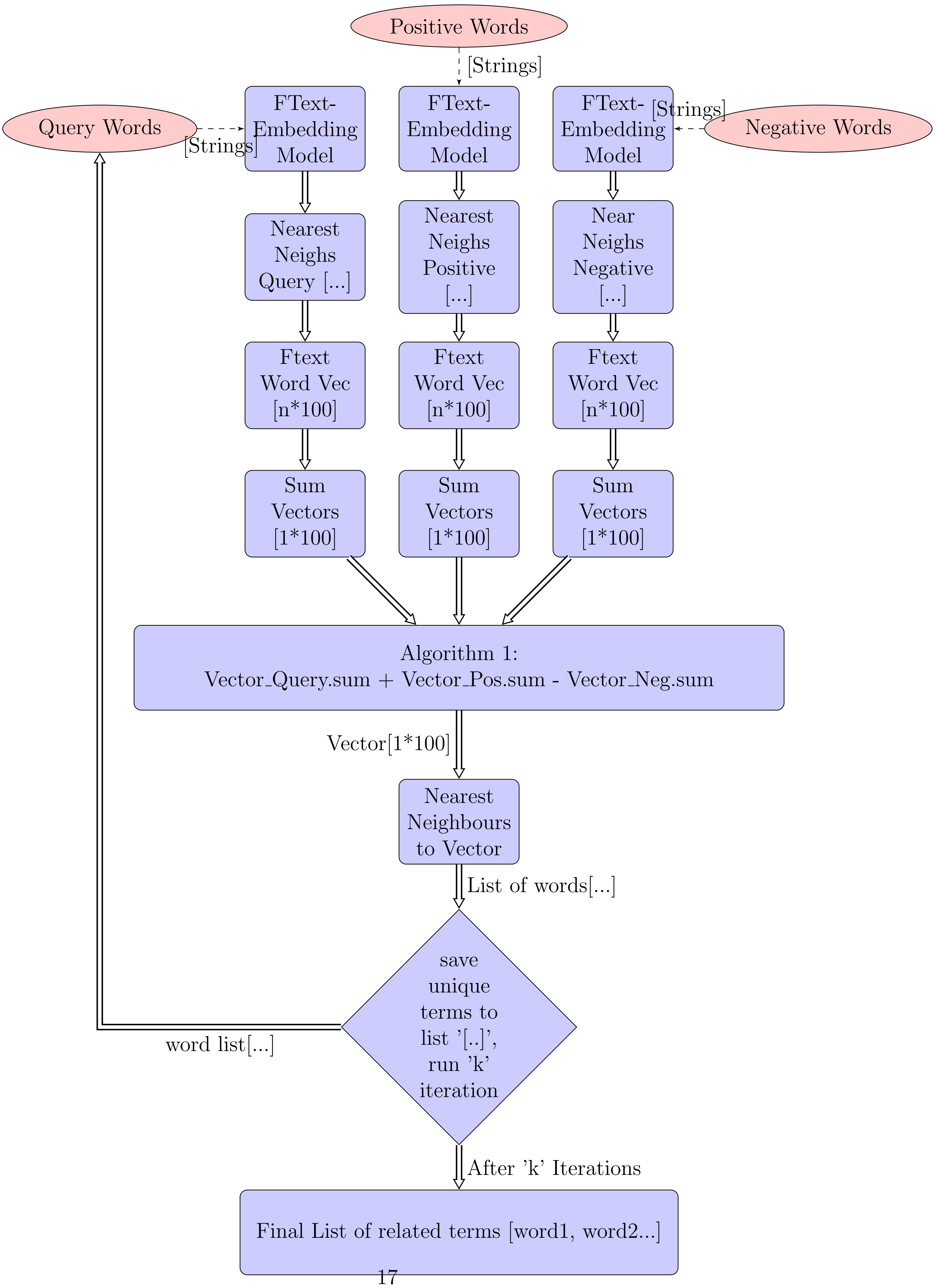
4.1.2 Flow Diagram


Positive Words
Query Words
FText-Embedding Model [1*100]
FText-Embedding Model [1*100]
FText-Embedding Model [1*100]
Negative Words
Sum Vectors [n*100]
Sum Vectors [n*100]
Sum Vectors [n*100]
Vector_Query.sum + Vector_Pos.sum - Vector_Neg.sum [1*100]
Nearest Neighbours to Vector lwords[...]
save unique terms to list '[..]', run 'k' iteration
List of words[...]
After 'k' Iterations
Final List of related terms [word1, word2...]

4.2 Algorithm-2

4.2.1 Process Flow

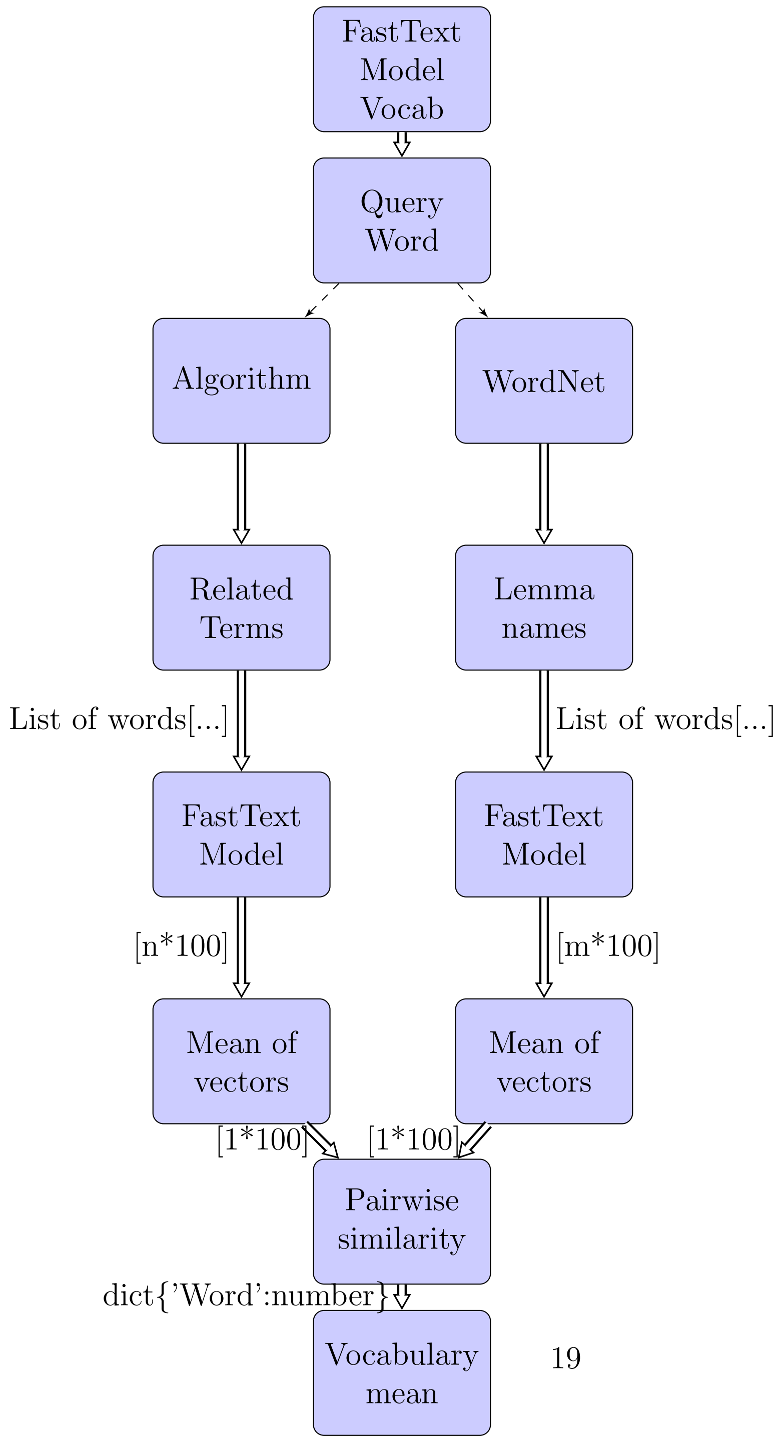
4.2.2 Flow Diagram



5

Validation

5.1 Flow Diagram



5.2 Process Flow

- Wordnet is used to verify the algorithm: All the related terms to a word are fetched into a list using all the lemma_names from the synsets for that word.
- The word vector[1*100] for the list of lemma_names [n] are fetched from the FastText model and concatenated into a matrix[n*100].
- The word vector[1*100] for the result_list [m] from algorithm are also fetched from the FastText model and concatenated into matrix[m*100]
- The corresponding average of each matrix are taken and the cosine distance between two are stored into dictionary for 1000 query 'word' taken from the FastText model vocabulary. Eg: ..., 'skiing': 0.9093364, 'sobbing': 0.90886176, 'dryer': 0.7378045,... (1000 query words took 4hrs, FastText has a vocab of 38 Million words)
- The mean of similarity 1000 the words are calculated to get overall efficiency of the algorithm, which results to 83%.

6

Conclusion

6.1 Research Aims

6.2 Research Objectives

6.3 Practical Implications

6.4 Future Research

6.5 Chapter Summary

Appendix A

(Appendix A title)

Appendix B

(Appendix B title)