申请上海交通大学学士学位论文

图论与深度学习结合的语义表示

论文作者 应松晟

学　　号 5130369019

导　　师 吕宝粮

副 导 师 Sabine PLOUX，赵　海

专　　业 信息工程（中法合作办学）

答辩日期 2017 年 7 月 13 日

Submitted in total fulfillment of the requirements for the degree of Bachelor in Information Engineering (Sino-French)

Semantic Representation with Graph Cliques and its Application in Machine Translation

Songsheng YING

Advisor

Prof. Baoliang LÜ

Co-advisor

Prof. Sabine PLOUX, Prof. Hai ZHAO

ParisTech Elite Institute of Technology

Shanghai Jiao Tong University

Shanghai, P.R.China

Jul. 13th, 2017

图论与深度学习结合的语义表示

摘　要

在自然语言处理相关研究中，分布式向量表示法是将单词信息编码的常用方法。这种表示方法克服了经典方法中以单词在词汇表中的位置进行编码带来的高维度问题，以连续空间中的一个实数向量为依托，并进一步提出了以向量度量为基准的潜在信息的计算方法。大量神经语言模型使用分布式向量表示法总结计算其系统内处理之文本的抽象信息。诸如 Word2Vec 和 GloVe 之类的分布式向量表示训练方法近年在学界大受欢迎。以此方法建立的词嵌入空间能保留语言学意义上的单词间规律和语义相对关系，因此这些词嵌入模型在自然语言测试中取得了惊人的成果。

这些以单词为基本单元的嵌入系统的缺点之一，在于语义和词语并非存在一一对应的关系。人类对于语言词汇的选择基于词义，而不是词形。为了建立更准确的语言模型，相比于单词，词义是嵌入连续空间实体的更优候选者。

词义嵌入的现有文献多采用受监督或半监督学习的方法。他们或依靠大量结构明确的手工数据库来构建一个语义网，要么需要人工对语义项进行聚类和规则编码。本论文提出了使用子元素相邻的完全子图（上下文词团 contexonym clique）来从语义层面对文本进行消歧义。我们利用从大型单语语料库中获取的单词词频和单词对同句出现频率等统计信息，构建以单词为顶点，以词和词间的相互关联性为边的图结构。为了获取随机单词的上下文词团集，我们通过对其构建上下文词组（contexonym）集合和以词间关系图结构为基准的图边抽取，从新生成的小型图结构中提取上下文词团集合。

在本论文中，上下文词团被视作词义的单元。这些上下文词团被用于对文本中的对应单词打上词义标记。词团的选择基于被消歧义单词实例所处的上下文语境：在上下文中出现的词条总集是被消歧义单词所具备的上下文词团集合选择词团的判断基准。此词义标记代表的语义含义是通过词上下文词团中的上下文词组体现的，该词团在连续语义嵌入空间中的向量与其他向量的空间关系也是词义含义的来源之一。

此论文使用基于 Skip-Gram 的训练方案，以被上下文词团标记从而被消歧义过的语料库为数据来源来学习语义嵌入表示。

我们使用英语词汇替代任务（Lexical Substitution）对基于上下文词团消歧义系统和语义嵌入系统进行了测试。评估结果显示我们的模型系统能厘清词义，并在嵌入系统中捕获到词义间的相关性。

关键词： 连续空间模型，图论，语义嵌入，词义辨析

Semantic Representation with Graph Cliques and its Application in Machine Translation

ABSTRACT

Distributed vector representation is a common method for Natural Language Processing researches to encode the word information into a continuous space. This representation method overcomes the dimensionality problem of classic one-hot word encodings, and presents latent calculable properties based on vector metrics. Neural language use the vector representation to further compute abstract information of the concerned text. Popular distributed vector representations such as Word2Vec and GloVe build an embedding space for words while conserving the linguistic regularities and semantic relational values. These models have achieved surprising results in NLP tasks.

A drawback for these word-based embedding system is that the semantic senses and the words do not form a bijection. Human language access is based on word sense, rather than word form. To propose more accurate language models, the word senses are better candidates to be embedded in a continuous space.

Existing works that embed senses are of supervised or semi-supervised methods. They either rely on an abundant well-formed database to construct a semantic net, or requires manual sense clustering and rule encoding works. This thesis proposes the usage of element-wise adjacent contexonym complete subgraphs (contexonym cliques) to semantically disambiguate the text. We make use of word occurrence and word-pair-wise co-occurrence statistics acquired from a large monolingual corpus. The statistics help building a graph structure where the words are vertices and the connection between words forms the edge. The contexonym cliques for one headword are extracted from the graph with the construction of a relevant contexonym set and a conversion of weighted edges to boolean edge.

In this thesis, we use contexonym cliques as the sense unit. The cliques are labeled to head-

words occurring in a context based on contextual word instances. This label disambiguates the text either by its clique member contexonyms, or the clique vector acquired from an embedding.

The embedding is learned using Skip-Gram based training scheme with a sense-annotated corpus. The corpus is labeled with cliques.

The entire system is tested with English Lexical Substitution Tasks. The evaluation results suggest the potential accuracy of the sense entity captured by the clique system.

KEY WORDS: Continuous Space Model, Graph Theory, Sense Embedding, Word Sense Disambiguation

Contents

Index of Figures

Index of Tables

Index of Algorithms

Chapter I

Introduction

1.1 Background

Natural Language Processing (NLP) is a discipline where natural languages are modelled to be representable and calculable by machine systems. The realated researches aim to convert texts into a programmable data structure while conserving the linguistic properties. While researchers have not yet a consensus on such a structure, driven by specific linguistic applications such as Information Retrieval, Machine Translation (MT), Human-Machine Dialogue, and the ever deepening involvement of Machine Learning / Deep Learning techniques on such subjects, the continuous space representation model is widely applied and has recently demonstrated surprising improvements against traditional models.

Contrary to conventional representations where each language unit is represented by an one-hot vector of the dimensionality of the size of the concerned vocabulary, the continuous space model converts language units (such as words, phrases) into high dimensional real valued vectors. This distribution technique is also called "embedding". Early attempts of embedding adopt structural methods, using cognitive processing techniques to embed words, such as works as WordNet [1], ConceptNet [2] and others who associate the lexicons with defined inter-relationship information. These dictionary-looking works rely on data sources of abundant linguistic information, they are thus expensive and time-consuming to obtain and update with daily evolving language.

Document level analysis could also be used to infer word distributed representation, there are works such as Latent Semantic Analysis (LSA) [3, 4]. To make use of token-level statistic information from a large corpus, a vocabulary-size dimensional matrix could be reduced on dimensionality by using mathematical methods like Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). The LSA assumption, that words of similar mean-

ings will occur in similar contexts, forms also the theoretical foundation of contexonym clique analysis [5], the famous word2vec [6, 7], and GloVe [8] embeddings. These models use word co-occurrences in a local context extracted from a large corpus document to deduce syntactic and/or semantic sense of the word.

The word2vec and GloVe methods demonstrate outstanding results on syntactic and semantic sense analysis tasks: using vector distance as a metric, the embedded vectors could be used to analyze word similarities and word analogies. However, such models presume that there is a bijection between words (or tokens in a more generalized terminology) and sense entities, which is not true given the existence of polysemies, homonyms and heteronyms. Table 1–1 shows that these embedding models trains word vectors that are close to synonyms of different sense items of the words.

Table 1–1 **word2vec and GloVe word similarities**

word	bat[a]
word2vec	bats, batting, Pinch hitter Brayan Pena, batsman, batted, Hawaiian hoary, Batting
GloVe	bats, batting, Bat, catcher, fielder, hitter, outfield, hitting, batted, catchers, balls
word	jaguar
word2vec	jaguars, Macho B, panther, lynx, rhino, lizard, tapir, tiger, leopard, Florida panther
GloVe	jaguars, panther, mercedes, Jaguar, porsche, volvo, ford, audi, mustang, bmw, biuck
word	appeal
word2vec	appeals, appealing, appealed, Appeal, rehearing, apeal, Appealing, ruling, Appeals
GloVe	appeals, appealed, appealing, Appeal, court, decision, conviction, plea, sought

[a] Bat being associated with the sport (catcher, hitter) and the animal (bats, batsman).

word2vec and **GloVe** mix different senses of words. These embeddings are trained from [6, 8], similarity using cosine distance. Examples selected from [9]

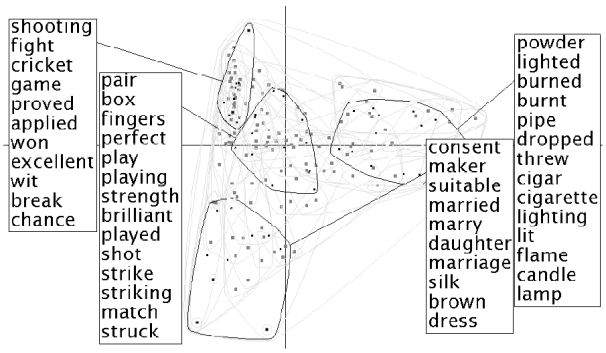
Word Sense Disambiguation (WSD) aims to amend this false presumption on word-sense bijection and to improve the accuracy of current representation models. Some continuous representation models mentioned above implement WSD such as WordNet-based methods. WSD does not necessarily rely on a continuous model, and various techniques have been exploited to deduce the sense of the word by referring to its context.

Dictionary or thesaurus based methods of WSD such as Lesk algorithm [10] matches the

dictionary definition against the instantial context of the word, later ML models analyze such matching information to train a classifier. Models based on graph (which is based on WordNet-like database) are also developped [11]. Another example of such kind is SensEmbed [12], which used a graph generated with linkage supplied by Babelfy [13] to disambiguate.

Semi-supervised methods such as Yarowsky [14], follows "one sense per collocation" and "one sense per discours" principles, use a small amount of hand selected seed word supplied with human-defined senses to train on a large corpus.

An example of unsupervised method is contexonym cliques. Contexonym cliques, which are extracted from word graphs linked by co-occurence weights, with the techniques of hierarchical clustering and Correspondence Analysis (CA) [15]. The extracted cliques are able to represent non-predefined separate different senses of a word. Each extracted clique has a set of clique words. After doing a CA, the clique entities and the related contexonyms are all transferred into one unique spatial representation. Figure 1–1 shows an example of a 2-dimensionally-projected, CA transformed space for cliques and contexonyms of the word "match", without further ado of analyzing clique properties, it is seen that the contexonyms of similar topics can be clustered together after CA, each region representing its own meaning. This method could, therefore also classify cliques which are believed to represent finer semantic units using similar methodologies.



The four clusters representing orderly the sense of "game", "partner", "marriage" and "match for lighting a fire"

Figure 1–1 **Clusters of contexonyms for "match"**

Clique extraction and enumeration, involved in this unsupervised method, is one of the 21

NP-complete problems. Enumeration is of exponential complexity in computation [16]. There are web pages [1] based on existing works [5, 17, 18] that calculate and display interactive clique and contexonym CA projections. Inquiries sent to this site could take more than 5 minutes to calculate associated cliques for one single word.

1.2 Project Proposal

While word embedding is a hot topic, sense embedding is a relatively few researched area.

A method to embed the nuances of words is researched in this thesis. The theoretical foundation of the method proposed is based on the capability of contexonym cliques to catch subtle semantic information, and embedding's empirically proven structural property of conserving linguistic regularities and other semantic information.

Early works on contexonym cliques analyze only the inter-relation for a small group of words, since the graph construction uses only the Point-wise Mutual Information (PMI) for directly co-occurring words without analyzing any second or third-level co-occurrences. word2vec embeds word into a high dimensional space without disambiguating word senses.

This work proposes firstly a WSD solution with contexonym cliques, then we use the WSD to generate a sense annotated corpus. With each content word in a context tagged with a clique, this model could associate the word with a nuanced sense.

This sense is measurable and comparable with the introduction of embedding. The embedding method for different word senses deduced from contexonym cliques follows training schemes derived from word2vec. The training uses the generated sense-annotated corpus as data source.

To evaluate the quality of the cliques and the embedded vectors, we propose using word similarity examinations and lexical substitution tasks. The word similarity could show the relativeness of the embedding method, the interactions between sense entities. The lexical substitution task could score this interaction.

Should the embedding be of sufficient quality and accuracy, we further propose its application in various NLP research directions including machine translation, so that the nuance could

[1] `http://dico.isc.cnrs.fr`

improve the translation accuracy and pertinence of the context.

The main contributions of the thesis are listed as follows:

- To capture the maximum co-occurrence information of content words, we apply firstly a lemmatization and a functional word removal on the corpus, which accumulates statistical information for fusional language terms and reduces the size of concerned lexicon.
- To extract efficiently all cliques (NP-hard) from the deduced graph, we use an adaptive filter to select the contexonyms of a given headword. To ensure a reasonable computation time and relevant clique extraction results, a dynamic adjustment mechanism is proposed to modify the graph to accelerate the calculation.
- The thesis proposes a tagging method to associate a word given a context with a pre-generated clique, so that the word could be disambiguated.
- Several training schemes are proposed to embed the cliques into a continuous space, a skip-gram based training on clique contexonyms, a doc2vec training applied on contexonyms and a modified word2vec model training simultaneously on clique and word vectors over a sense-annotated corpus.

1.3 Thesis Structure

1. Chapter II describes the notion and the generation of contexonym cliques.
2. Chapter III introduces the process of labeling a word within a context with contexonym cliques.
3. Chapter IV gives three models to embed contexonym cliques into a distributed vector space.
4. Chapter V narrates the experiments and evaluation tasks performed on our models.
5. Chapter VI gives a possible navigation from currently obtained results to practical applications such as machine translation.

Chapter II

Graph-based Contexonym Clique Extraction

2.1 Definitions

Definition 2.1 (Cliques)**.** In an undirected graph, a clique is a subset of vertices that within its induced subgraph, every two distinct vertices are adjacent.

Definition 2.2 (Contexonyms)**.** A token is a contexonym of another token if these two tokens co-occur in one same unit of text concerned. The unit could be a context window, a phrase, a paragraph or a document.

Definition 2.3 (Contexonym cliques)**.** In a graph where each vertex represent one unique token of a particular vocabulary, each edge connecting two vertices is weighted by the PMI score of the two corresponding tokens (calculated in one particular corpus), the contexonym cliques are the cliques extracted from the same graph as the one mentioned above, except that some edges of inferior weight than a context-aware threshold are disconnected, and that some vertices not meeting a set of criteria are removed from the graph.

2.2 Introduction

Former works on synonym cliques and contexonym cliques have reported a continuous transition of word senses [5, 18]: if we first sort the vertices (representing tokens) in cliques alphabetically, then sort the cliques, we could often infer different semantic values from the words contained in cliques. In Example 2.1, we selected some contexonym cliques of the word *"good"* extracted from the English Wikipedia using our method, to show the transition from a event-descriptive value, a romance-related value to a general verbalized value of *"good"*. The entirety of cliques is listed in appendix A.

Example 2.1 (Selection of contexonym cliques of *"good"* extracted from Wikipedia)**.**

10. accident, crash, disaster, fatality, tragedy

11. accident, crash, disaster, terrible, tragedy

282. cyclone, deadly, disaster, fatality, flood, flooding, tornado

283. cyclone, deadly, fatality, flood, recorded, tornado

58. affected, cyclone, disaster, flood, flooding, tornado

54. affected, bad, disaster, drought, famine, flood, flooding

66. affection, boyfriend, classmate, crush, girlfriend, jealous

17. acquaintance, boyfriend, classmate, co-worker, crush, friend, girlfriend

18. acquaintance, boyfriend, classmate, co-worker, crush, friend, neighbor

40. acquaintance, friend, intimate, lover, stranger

63. affection, beautiful, beloved, jealous, lover

93. amazing, beautiful, brilliant, exciting, it's, pretty

96. amazing, brilliant, feat, remarkable

97. amazing, brilliant, remarkable, truly

98. amazing, exciting, funny, it's, pretty, really, stuff, thing

These fine-grained transitions could also be grasped by Fine-grained Lexical Knowledge model proposed in [19], where such relations are named *near-symonymy*, and it is based on a large amount of linguistic sources such as *Webster's New Dictionary of Synonyms* and other compiled dictionaries. However, with synonym cliques which are produced in an unsupervised fashion, the cliques are also used as indicators for the tokens' precise semantic values.

Compared with Yarowsky's method [14] (which is semi-supervised as introduced in Section 1.1), the contexonym cliques do not predetermine the number of senses and thus are more autonomous and are usually of finer granularity, provided a dense or a large word association table.

Based on the clique theories, Ploux et al. introduced in [18] a mapping method for two clique spaces of different languages. The evaluation in the work shows that synonym cliques could be used to suggest translation candidates. This suggests a property to be exploited between clique sets of different languages. The sense captured could be calculated in one same environment given a proper algorithm.

Although symmetry and transitivity exist only in synonyms but not in contexonyms, and contexonyms are usually of various grammatical components, we hypothesize that a contexonym cliques could capture semantic information based on the results reported in [5], and thus use contexonym cliques as a data source in the labeling step and the embedding step of the experiment.

2.3 Clique Enumeration Preperation

2.3.1 Lemmatizing the corpus

For the case of many synthetic languages such as the Romance family and the Germanic family, and some analytic languages such as English, some morphemes such as prefixes and affixes could be joined to a *root* to create various forms of words that can be used accordingly to the number, time, gender, voice, etc in a given situation. Sometimes the variations of the *root* do not even necessarily conserve its basic form. In Example 2.2, there are 3 *roots*: “peu-”, “pou-”, “pu-”.

Example 2.2 (Variations of the French verb *“pouvoir”*)**.** peux, peut, pouvons, pouvez, peuvent, pouvais, pouvait, pouvions, pouviez, pouvaient, pu, pus, put, pûmes, pûtes, purent, pourrai, pourras, pourra, pourrons, pourrez, pourront, pourrais, pourrait, pourrions, pourriez, pourraient, puisse, puisses, puissions, puissiez, puissent, pusse, pusses, pût, pussions, pussiez, pussent.

This gives an important of fragmentation of vocabulary entries, if we leverage the granularity of semantic value kept in inflections such as gender and voice against the abundance of textual data, to capture semantic information over syntactic values, it would be better to unify the inflections onto the one common “root” lemma of the variations.

In this thesis, we use TreeTagger [20, 21] to label tokens in the corpus with Part-of-Speech (POS) tags and lemmatize them. TreeTagger supports multiple languages and there are language configuration files available online, each supplied with its tag set. Take English parameter file on TreeTagger’s page [1] for example, it has 36 different tags (see Table 2–1). We discard tokens of

[1] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

22 tags being marked as functional words (and symbols). For the rest we substitute the original word with its lemma supplied by TreeTagger.

Table 2–1 **English tagset [22]**

Tags		
TAG	POS	Discard
CC	Coordinating conjunction	×
CD	Cardinal number	×
DT	Determiner	×
EX	Existential *there*	×
FW	Foreign word	×
IN	Preposition or subordinating conjunction	×
JJ	Adjective	√
JJR	Adjective, comparative	√
JJS	Adjective, superlative	√
LS	List item marker	×
MD	Modal	×
NN	Noun, singular or mass	√
NNS	Noun, plural	√
NP	Proper noun, singular	√
NPS	Noun, plural	√
PDT	Predeterminer	×
POS	Possessive pronoun	×
PP	Personal pronoun	×
RB	Adverb	√
RBR	Adverb, comparative	√
RBS	Adverb, superlative	√
RP	Particle	×
SYM	symbols	×
TO	*to*	×
UH	Interjection	×
VB	Verb, base form	√
VBD	Verb, past tense	√
VBG	Verb, gerund or present participle	√
VBN	Verb, past participle	√
VBP	Verb, non-3rd person singular present	√
VBZ	Verb, 3rd person singular present	√
WDT	Wh-determiner	×
WP	Wh-pronoun	×
WP$	Possessive wh-pronoun	×
WRB	Wh-adverb	×

As functional words do not have substantial semantic value in the context unit, and proper nouns usually have one-to-one mapping with its sense, after applying the token filtering against POS labels and lemma substitution, we now obtain a corpus with *sentences* composed by content word lemmas. In addition, the uppercase letter as the head of a sentence is also put to lowercase if appropriate. This corpus is called from now on **lemmatized corpus**.

Note that TreeTagger has problem with tagging certain POS labels, furthermore, for heteronyms and homonyms, it could produce lemmas containing all related senses. An example is the French word *"fil"* (string, voltage line, train of thought) and *"fils"* (son, descendant, singular/plural), which would produce one same lemma *"fil|fils"* as the tagger could not distinguish the two. The English word "tear" is also an example of heteronym.

2.3.2 Statistics

Given the lemmatized corpus, we now obtain the occurrence statistics for each lemma by iterating through the corpus. We take only the 60 000 or so most frequent lemmas into consideration, with the rejection the most 100 lemmas. (This analysis for this selection is detailed in Section 5.1.1.) Then we obtain co-occurrence count of lemmas with the language unit set as a lemma, and the context environment set as a phrase. The co-occurrence count for a pair of distinct lemmas is the product of their occurrence counts in one sentence, so that stronger connections inferred by multiple co-occurrences between lemmas could also be taken into account.

2.3.3 Co-occurrence graph

For one given lemma w_i, we have its occurrence count $fr(w_i)$, and its co-occurrence count with another lemma w_j $co(w_i, w_j)$. In an undirected graph where each vertex is a lemma occurring in the lemmatized corpus, we connect the vertex representing w_i and w_j with an edge of weight $\frac{co(w_i,w_j)}{fr(w_i)\times fr(w_j)}$, which is the PMI score by definition.

We repeat the process above for all lemmas in the vocabulary. Once the co-occurrence graph obtained, we adjust the connections in the graph with other criteria: we disconnect the edges where the condition $\{\frac{co(w_i,w_j)}{fr(w_i)} \leq \alpha\} \bigvee \{\frac{co(w_i,w_j)}{fr(w_j)} \leq \alpha\} \bigvee \{\frac{co(w_i,w_j)}{fr(w_i)} \geq \beta\} \bigvee \{\frac{co(w_i,w_j)}{fr(w_j)} \geq \beta\}$ is

met. This filter configuration filters noises in the corpus, and gives the most relevant contexonym candidates during the development, as α and β are tuned. Note that we do not yet filter on PMI score, as the PMI filter is applied at a later step for clique extraction acceleration.

We now have a graph with vertices connected, and we have removed weak and over-strong connections.

2.4 Enumeration of cliques

2.4.1 Selection of vertices for clique extraction

The original contexonym clique extraction [5] constructs two association tables to construct a boolean-edge graph. We follow the basis of such method, except that we have preliminarily filtered the graph beforehand, and that we render the association more flexible and dynamic using adaptive parameter adjustment.

Our goal is to generate up to hundreds of cliques for each polysemous lemma, each clique being composed by up to 15 lemmas. Therefore, the association table should be composed of relevant and very representative lemmas associated with the target lemma.

Once the target lemma is selected (to which the cliques belong), we select first γ percentage of its connected lemmas (children) sorted by edge weight, be it $w_i : c_{i,1}, c_{i,2}, \ldots, c_{i,floor(\gamma \times \#\{connectedlemmas\})}$. For each $c_{i,j}$, we then add its own first δ connected lemmas (grandchildren) to the selection. When adding the grandchildren to the selection, if the headword (selected lemma to begin with) is not in the list of grandchildren, we remove this grandchildren set and its corresponding child lemma from the selection. The retro-removal of lemma is another assurance of the pertinence of the contexonyms.

γ and δ are initialized with two values that are determined during the development phase, the selected lemmas will be the vertices in the subgraph from which we will extract cliques.

2.4.2 Enumeration

As stated above we wish to obtain cliques containing up to 10 - 15 lemmas each, and we do not want a over-fragmented sense collection: we could adjust the size of the subgraph by selecting

the contexonyms with dynamic parameters γ and δ (see Section 2.4.1), and adjust the density of the graph by adjusting the threshold ϵ on edge weight (as in Section 2.3.3 we have not yet filtered on PMI).

With selected contexonyms and the threshold which is prefixed to a supplied value, the subgraph is generated from the co-occurrence graph: the vertices are selected contexonyms, and the connected edges are those of weight superior than the threshold, being set simply to *True*. We will calculate the size and the number of connected edges, and adjust the parameter accordingly to the latest result of calculation and re-generate the association table, then a more appropriate subgraph in terms of size and density.

γ and δ (used to select child lemma and grandchild lemma) and the threshold ϵ are adjusted in a multiplicative way. These configurations were tested to work efficiently in experiments. Typically this self-adjustment is crucial since monolingual contexonym graphs are rather dense, which could introduce an expensive time comsuption. In the experiment with English Wikipedia, the association table is typically reduced 2 times and the threshold ϵ is augmented more than 4 times for popular lemmas.

We use an algorithm based on Bron and Kerbosch's [23] to enumerate cliques (extract all possible cliques from the subgraph). A modification is made based on [24], since removing the need to traverse adjacent vertex will not influence the enumeration in the recursion tree (see Fig 2–1). Then we added a heuristic on choosing the vertex with the largest (untraversed) connected edges, hopefully to reduce the depth of the recursion tree. The theorems based on which these modifications are made are demonstrated in [24].

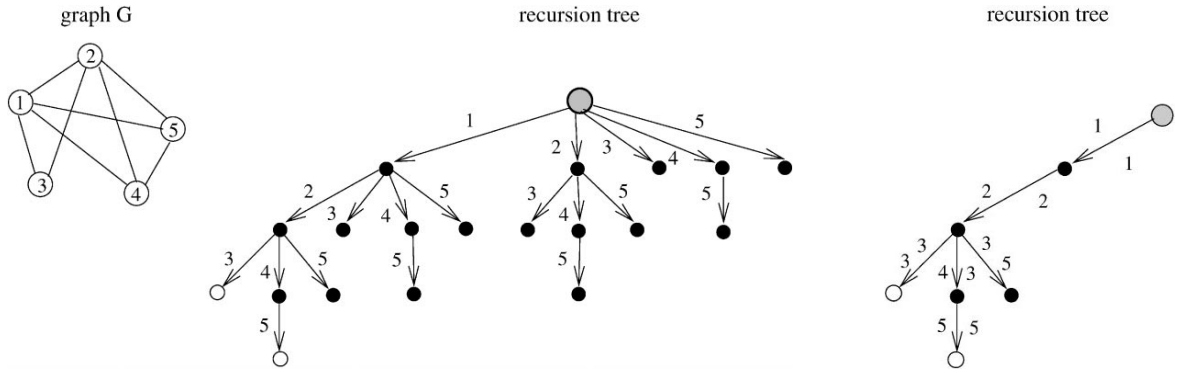
This algorithm used in the experiment is described in Algorithm 2–1. Despite the improvement, the worst case of time complexity is still exponential. Limiting the number of edges and vertices reduces enormously the time consumed for enumerating all cliques. On the test machine where we use one Intel Xeon core of 3.33 GHz, the average time to enumerate all cliques from a graph less than 2000 edges and 2000 vertices is less than 4 seconds. The reported cliques during the algorithm's execution are collected in a specific data structure where cliques of one headword (lemma) is easily found and compared with each other.

In Appendix A we provide the contexonym cliques for *"good"* as an example. We could

Algorithm 2–1 Modified BK's algorithm with heuristic to enumerate all cliques in a graph $G = (V, E)$

C: set of vertices belonging to a possibly non-complete clique
P: set of vertices that can be added to C
S: set of vertices which is not joinable to the currently forming clique
N: set to store neighbors for a vertex
ENUMERATE(C, P, S):

1: $P \leftarrow \{c_1, c_2, \ldots, u_k\}$ (all vertices of G)
2: $C \leftarrow \emptyset$
3: $S \leftarrow \emptyset$
4: **if** $P = \emptyset$ **then**
5: **if** $P = \emptyset$ **then**
6: REPORT FOUND CLIQUE
7: **end if**
8: **else**
9: $c_s \leftarrow \arg\max_{\forall c_i \in V} \#\{v \in V \mid \{v, c_i\} \in E\}$
10: **for** $i = 0 \rightarrow k$ **do**
11: **if** $\{c_i, c_s\} \notin E$ **then**
12: $P \leftarrow P \{c_i\}$
13: $P \leftarrow P$
14: $S \leftarrow S$
15: $N \leftarrow \{v \in V \mid \{v, c_i\} \in E\}$
16: ENUMERATE($C \cup \{c_i\}$, $P \cap N$, $S \cap N$)
17: $S \leftarrow S \cup \{c_i\}$
18: **end if**
19: **end for**
20: **end if**



Each edge in the recursion tree is the inclusion of the labeled vertex from the graph into the candidate collection. While the first recursion tree searches if vertices 1 and 3 form a clique, the next recursion tree does not since 3 is adjacent to 2 so the possibility of forming a clique is examined anteriorly when vertex 2 is traversed.

Figure 2–1 **Recursion tree for original and modified Bron and Kerbosch's algorithm [24]**

infer from the variations of the clique words that the cliques tend to catch minute semantic differences.

Chapter III

Labeling Words in a Given Context

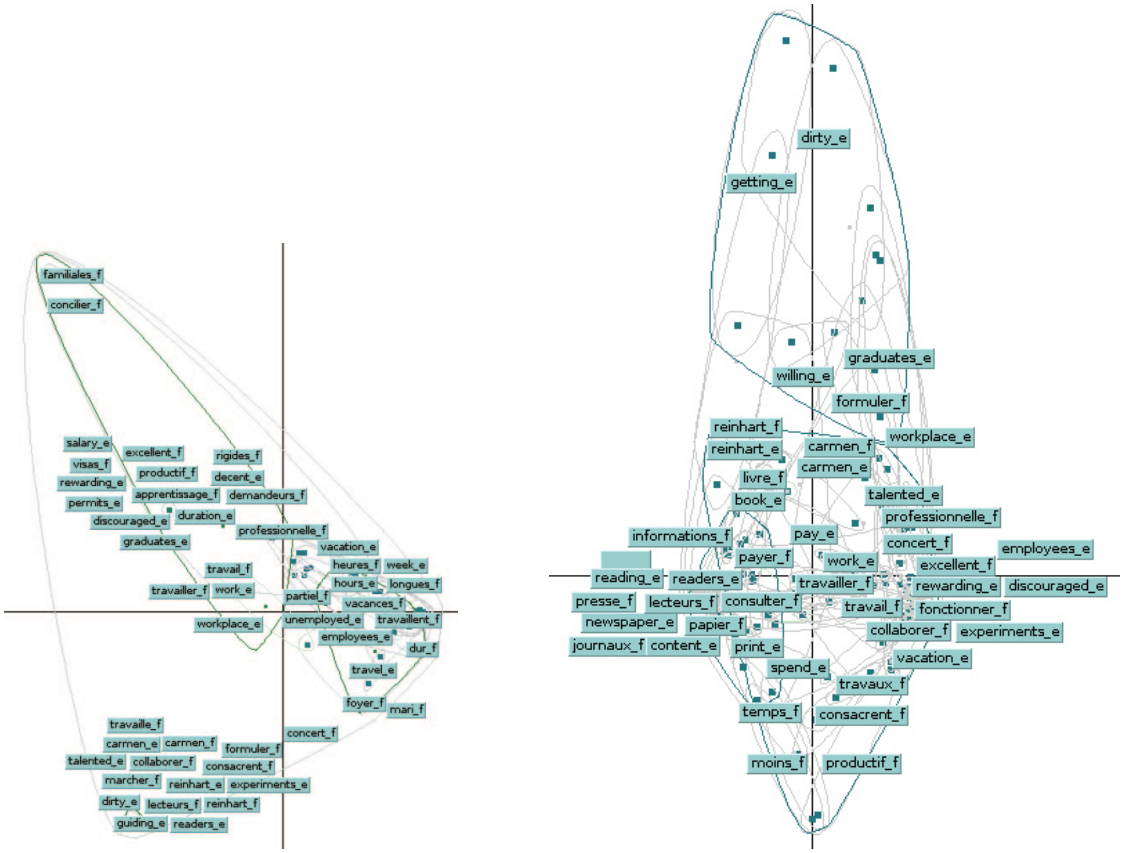
3.1 Introduction

In former works on semantic representation with cliques [5, 17, 18], cliques for one headword are usually analysed with Correspondence Analysis. The cliques are projected with CA to a 2-d plan, to show the distances and a clustering, and categorize the semantic value represented by each cluster with clique member elements. In a matrix where columns are tokens and rows are cliques, the boolean matrix is populated with $\Big[(contexonym)_j \in (clique)_i\Big]$. The CA on the clique matrix determines a set of basis of the linear space that gives the maximal spatial variance. After applying the change of basis to the matrix, we select the first two dimensions of the matrix space, and project the transformed row and column vectors (which are respectively cliques and tokens) onto the geographical space. The spatial distance is most explicitly displayed in this fashion.

The distance on the projected space gives a visual indication on the relations between cliques and tokens, the actual metric to evaluate the similarity is χ^2 distance (of vectors from the original clique matrix). The reason to adopt χ^2 distance as metric is detailed in Section 3.3.

An application of this visualization, is that we find that if we merge the clique matrix with another context word's clique matrix, the CA projected space would have cliques and tokens related to this context around the origin point in the projected space. In the newly merged matrix, the cliques sharing contexonyms from two matrices tend to have less variations than other cliques, the other cliques are therefore pushed CA away from the $(0, 0)$ point. The method of joining another clique's information is discussed in [5], where JI et al. used a contexonym merging method to extend the set of contexonyms (see Fig 3–2). An extended example (Fig 3–1) is taken from [17], where the tokens are bilingual (French/English) words.

JI et al. also propose using *linking contexonyms* to label the headword's sense based on



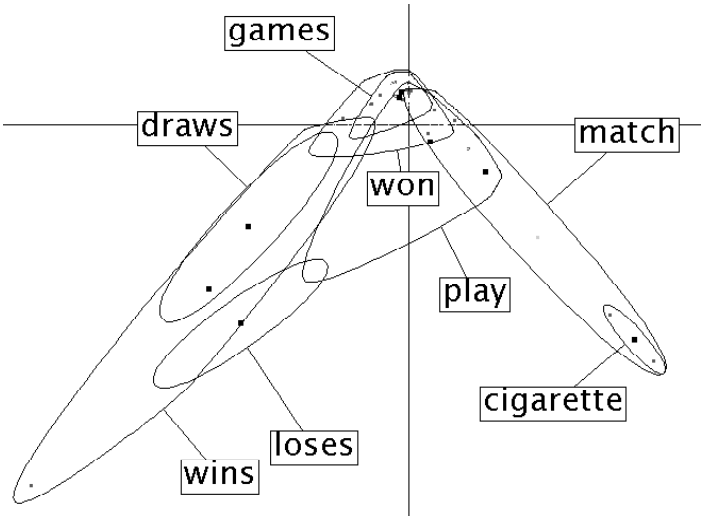
Each point in the figures is a clique, each box of word is positioned with its center at the projected position for corresponding word (but minimally moved to avoid overlapping). The lines are the cluster boundary where similar cliques are grouped [18]. The colored borderlines are the cluster borders for the concered headword's clique, not the context's. In the left we see 3 major clusters of cliques, each representing "employment", "labour" and "achievement, piece of art". With the addition of the context, we see that the cluster located in the left-down side in the left figure is placed near the origin in the right figure, the cliques in which are most relevant to the context.

Figure 3–1 **CA projected space for the English word *"work"* with and without context *"readers"*[17]**

the linking contexonym of another headword in the phrase. For example, in Fig 3–2, the sense selection of "match" is a decision based on the shared contexonyms *play* and *games* of the headword "wins" in the context.

Furthermore, we could also consider the contexonyms as seed words in Yarowsky's [14] method, and build a decision tree to add more contexonyms.

Labeling a word in a sentence with contexonym cliques however, is not as intuitive as adding a selected context. The linking method of Ji et al. requires a preliminary knowledge of the representative contexonym. The selection of headword "wins" is smart. For machines without enough linguistic knowledge, the dumb way of trying to merge every headword's in the context is time-consuming. Furthermore if in a context window multiple headword presents multiple common contexonyms representing different semantic values, the system does not have a good criterion to make a choice.



The clique matrix of "wins" and "match" are merged and projected by CA. The CA pushed the "game" value of match to the centre.

Figure 3–2 **CA projected space *"merge"* with context *"wins"***

In this work, we do not represent cliques graphically, however we could still measure the distance between cliques using metrics. The observation from the contextual information sup-

plement experiments shown above suggest that adding new entries in the clique matrix could change drastically the CA projected space. Similarly with the introduction of new cliques into the matrix, it would change the ranking of distances between clique pairs. Especially, with the objective of labeling a word out of its context, if we could add a pseudo-clique of the word to label, which is formed apart from the context, to the matrix, we could associate a word occurring in a sentence with a specific clique, given the distances between cliques and the added pseudo-clique.

A headword has 56.1 contexonyms in average. The generation of pseudo-cliques is rarely successful since the contexonym set size is limited, the overlapping of contexonym and a context word is relatively rare. At the generating time of a pseudo-clique from the lemmatized sentence, they do not usually share many lemmas with the set of contexonyms of the target word. This is a direct result of adjusting subgraphs when enumerating cliques, as the graph size is limited, and a lemma could have much more related contexonyms than those already in the subgraph. This lack of contexonyms could also be due to the property that cliques are a shallow method (contrary to deep ones), which do not investigate deep relationships between the lexicons.

To solve the potential data sparseness problem [25], in this chapter, we propose a new method to add consistently contexonyms, using word similarity as a metric, which is more light, as it does not build extra dedicated structure, and more efficient, as it does not search against all contexonyms of all lemmas, and select among the common contexonyms and lemmas deciding which to use for the context.

3.2 Expanding the contexonym list

With the success of word2vec and GloVe as introduced in Chapiter I, we make use of this kind of un-ambiguated word embedding to roughly estimate the similarity of contexonyms. The similarity is used to activate contexonyms in the clique set, so that a pseudo-clique could be generated. The word embedding is trained on the same lemmatized corpus as when we extract the cliques, following standard Continuous-Bag-of-Words model (see Fig 4–3).

The activation of current contexonym is linked with a threshold ζ: one contexonym is marked as occurring if it has at least one lemma occurring in the lemmatized sentence, whose similarity

with the contexonym is above the threshold.

It is a dynamic process, as it do not modify the matrix structure, nor does it require CA manipulations, and the extension is called at execution time. The selection is based on the properties exploited for the headword. The metric of similarity is cosine distance, as it is used as default in distributed vector embedding works such as word2vec and GloVe.

3.3 Clique selection based on distance

The contexonym expansion increases the probability of a multi-contexonym pseudo-clique being generated from the context. Then χ^2 distances between this pseudo-clique and all other cliques for the target headword are calculated, and used as the metric to attribute nuanced semantic information to the headword.

The reason to use χ^2 distance is that unlike distributed word vectors, using contexonyms as dimensions are not as homogenous as a fixed distributed model. Contexonyms could be inter-related, thus if a lemma is close in the trained embedding to more than two contexonyms, it could activate a lot contexonyms. Conversely, if a contexonym is activated by a lot of lemmas, this contexonym is not as definitive as other contexonyms which are less likely to be activated in terms of defining a contextual environment. To make amend for the unbalance weight of each column of the matrix, χ^2 distance (see Equation 3–1) captures the *selectivity* of each column and modifies the distance based on it.

$$\chi^2(clique_a, pseudo-clique_p) = \sum_{k=1}^{n} \frac{\sum_{i=1}^{n}\sum_{j=1}^{p} x_{ji}}{\sum_{i=1}^{p} x_{ik}} \times \left(\frac{x_{ak}}{\sum_{i=1}^{n} x_{ai}} - \frac{x_{bk}}{\sum_{i=1}^{n} x_{pi}} \right)^2 \quad (3\text{–}1)$$

A good example is the notion of functional words like “the” in English, “un”, “une”, “le” in French. They are among the most frequent words in each language, while they are not as “selective” as content words. If a functional words appear in the matrix columns, the pseudo-clique containing the functional words could be matches against many cliques in the matrix, though those matches are not selective at all. Although functional words are removed in former steps, this phenomenon of sub-selectivity is still to avoid with appropriate distance metric.

Example 3.1 (The partial clique matrix of *"house"*)**.** In Table 3–1, we listed the contexonyms of the English word "house" in order of their column index in the clique matrix. We marked with bold font the first appearance of lemmas not in the set of contexonyms. There are in total 38 lemmas outside the contexonym set and 60 lemmas in the set that can activate a contexonym. Then a part of the clique matrix is displayed in Equation 3–2, where n is the number of contexonyms, p is the number of cliques (including the pseudo-clique as the last clique in rows).

$$\begin{array}{c} \\ cli_1 \\ cli_{15} \\ cli_{21} \\ cli_{66} \\ cli_{96} \\ \dots \\ cli_p \end{array} \begin{array}{c} \begin{array}{cccccccc} con_0 & con_1 & con_2 & con_3 & con_4 & con_5 & \dots & con_n \end{array} \\ \begin{pmatrix} False & False & False & True & False & True & \dots & False \\ False & False & False & False & False & False & \dots & False \\ True & False & False & False & False & False & \dots & False \\ True & False & False & False & False & True & \dots & False \\ True & False & True & False & True & False & \dots & False \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ False & False & False & False & False & False & \dots & False \end{pmatrix} \end{array} \tag{3–2}$$

Table 3–1 **Contexonyms and its Similar Words**

Col ID	Contexonyms	Similar lemmas ($\zeta = 0.8$)	Similar lemmas ($\zeta = 0.7$)
1	book	**author**	**foreword**, **essay**, **biography**, non-fiction, **novel**
2	newsletter	∅	quarterly
3	deal	∅	**contract**
4	nonfiction	non-fiction	∅
5	newspaper	∅	∅
6	editor	∅	editorial, editor-in-chief, **editorship**
7	magazine	∅	**magazine's**
8	London-based	∅	∅
9	editor-in-chief	∅	editor
10	editorial	∅	editor
11	open-access	∅	∅

Table 3–1 – *Continued on next page*

Table 3–1 – *Continued from previous page*

Col ID	Contexonyms	Similar lemmas ($\zeta = 0.8$)	Similar lemmas ($\zeta = 0.7$)
12	pamphlet	∅	∅
13	IDW	∅	∅
14	merchandising	∅	**merchandise**
15	bookseller	∅	**stationer**, bookselling
16	self-publishing	∅	**print-on-demand**
17	bookselling	∅	bookseller
18	bookshop	∅	bookstore
19	licensing	∅	**license**
20	bookstore	∅	bookshop
21	non-fiction	nonfiction	book, author
22	magnate	∅	**tycoon**
23	desktop	∅	∅
24	fiction	∅	**science-fiction**, novel
25	copyright	∅	∅
26	comic	∅	**comic-book**
27	children's	∅	∅
28	venture	∅	∅
29	print	∅	∅
30	bestseller	**best-seller**	∅
31	conglomerate	∅	∅
32	printed	∅	∅
33	printer	∅	printing
34	royalty	∅	∅
35	printing	∅	printer
36	imprint	∅	∅
37	journal	∅	publication, quarterly, periodical
38	scholarly	∅	**scholar**
39	journalism	∅	**journalistic**
40	encyclopedia	e**ncyclopaedia**	**encyclopedic**, **dictionary**
41	publication	∅	periodical, **article**, journal, **published**

Table 3–1 – *Continued on next page*

Table 3–1 – *Continued from previous page*

Col ID	Contexonyms	Similar lemmas ($\zeta = 0.8$)	Similar lemmas ($\zeta = 0.7$)
42	multimedia	**multi-media**	**interactive**, **audiovisual**
43	catalog	**catalogue**	∅
44	monograph	∅	**two-volume**
45	textbook	∅	∅
46	company	**subsidiary**	**company's**
47	hardcover	paperback, **hardback**	**softcover**
48	reprint	∅	∅
49	publisher	publishing	∅
50	publishing	publisher	∅
51	quarterly	∅	**bi-monthly**, journal, newsletter
52	house	∅	mansion, cottage
53	graphic	∅	∅
54	media	**medium**	∅
55	periodical	∅	publication, journal
56	paperback	hardcover	hardback
57	dissemination	∅	∅
58	peer-reviewed	∅	∅
59	literary	∅	**literature**, **poetry**, **poet**, **writing**
60	tabloid	∅	∅

Using Equation 3–1, we could calculate the relative distance between each clique and the pseudo-clique. The closest clique is labeled to the headword. This disambiguation could later be used as additional information in other NLP tasks such as MT, Knowledge Acquisition etc..

Chapter IV

Sense Embedding with Contexonym Cliques

4.1 Introduction

In this step, we are to assign to each clique (of all lexicons) a vector in an embedding space. This step is named Sense Embedding since we consider contexonym / synonym cliques as a fine unit of semantic sense.

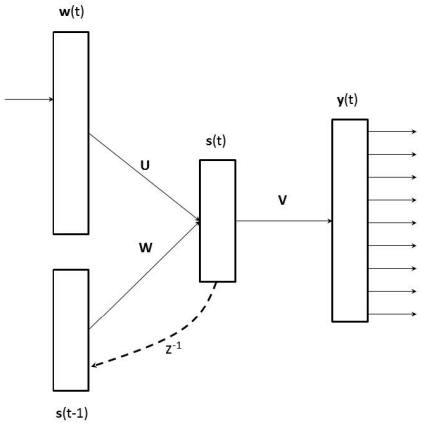
From the perspective of terminology, Sense Embedding is evolved from Word Embedding, since currently *sense* is all treated as a sub-unit of *word*. Once embedded, we could hopefully establish the notion of "similarity" in a formally defined way. We developed 3 procedures to calculate the sense (clique) vectors deriving from the two word2vec models: Skip-Gram and Continuous Bag of Words (CBOW).

4.1.1 Word Embeddings

Before Word Embeddings (WE) come into place, the NLP tasks used widely the N-gram models. Hinton et al.'s distributed representation of words [26] improved neural network based language models' performance over traditional N-gram models.

Mikolov et al. developed a *word2vec* toolkit, which is one of the most popular word embedding generation and evaluation toolkits. Firstly, they used a Recurrent Neural Network Model (see Fig 4–1) [6], use a matrix of weights to project one-hot input words onto a fixed-dimensional vector to predict the next word. This model is based on former works such as Bengio, Ducharme and Vincent's [27], which uses former input words to predict the next word in the sequence. As a result, they find a multiple level of similarity that exists among words: syntactic and semantic similarities. Tests reveal that this RNN version of word2vec outperforms LSA [4] and other traditional models.

Later, Mikolov. et al. propose new log-linear models [7] to minimize computational com-



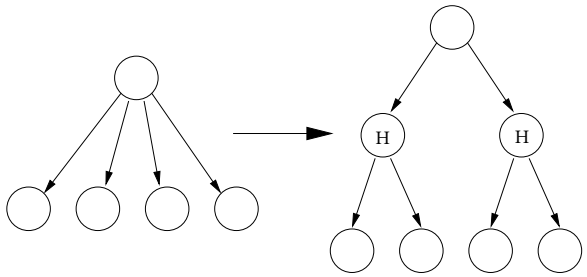
Each incoming word encoded by one-hot vector is then projected onto a lower dimensional space, then its addition with weighted historical status updates the hidden status. The hidden status is then transformed by a softmax to distribute the probability of activation of each word (a vector of dimensionality equaling the size of vocabulary, same as the input one-hot vector).

Figure 4–1 **RNN model used to embed words**

plexity of building such an embedding, while maintaining a high accuracy. These two models are CBOW (see Fig 4–3) and Skip-Gram (see Fig 4–4). CBOW is a neural network model which uses surrounding words to predict the surrounded word, while Skip-Gram predicts surrounding words given a central word. The notion of hierarchical softmax introduced by Morin and Bengio in [28] is adopted by Mikolov et al. in [29] to accelerate the convergence during training.

A classic Hierarchical Softmax implementation in word vector estimation training is to use binary trees to organize the leave nodes each representing a word in the vocabulary (see Fig 4–2). Each internal node connects two child nodes with relative probabilities. Therefore, when projecting a particular word's vector onto the vocabulary space, the distributed probability among the words is defined by random walks iterating through the binary tree. Moreover, the evaluation amount required to learn the output layer is reduced from V, the size of the vocabulary (lexicon more generally), to $\log_2 V$, since updating one word involves updating all the nodes along the traversing path, which is also the parent node of other words. Driven by this concept, the toolkit's implementation is to build a Huffman tree of vocabulary words, as it requires least updates, and less-frequent words tend to get a more accurate representation as its parent nodes

are updated more often.



The probability distribution over the set of vocabulary words is organized in an hierarchical fashion, so that the probability of each word is the product of the probability of all nodes on the path from root to the leave node.

Figure 4–2 **Hierarchical Softmax**

Another improvement made in [29] is the heuristic introduction of sub-sampling of the frequent words. This is an amendment regarding to the overly-frequent words: if those words occur in too much contexts, the errors detected when a nother word is projected onto these words are not as crucial as other words. This represents the similar notion of "selectivity" as χ^2 distance is introduced. The invented probability of discarding a certain word is given by Equation 4–1, where threshold is typically around 10^{-5}.

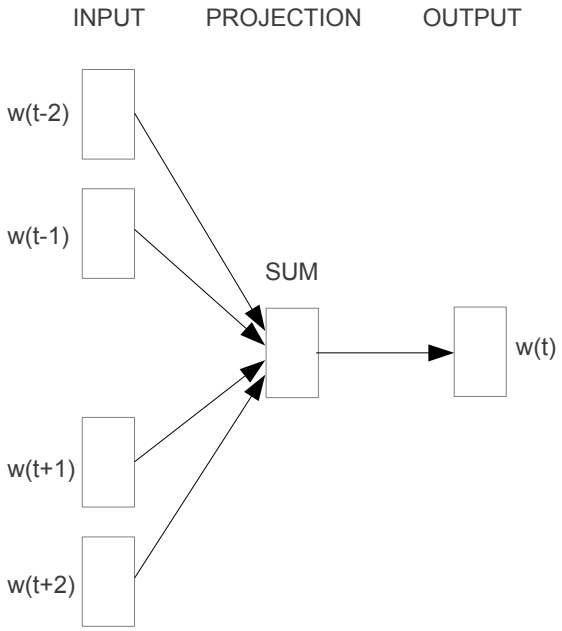
$$\mathbb{P}(w_i) = 1 - \sqrt{\frac{threshold}{fr(w_i)}} \tag{4–1}$$

This sub-sampling configuration brought 19% correctness improvement of semantic value evaluation on Huffman Hierarchical Softmax model over other models.

These word embeddings has achieved surprising results in word similarity and relational analogy tests.

4.1.2 Other Sense Embeddings

Two other models that introduce sub-word semantic unit into embedding are inspiring to our work.



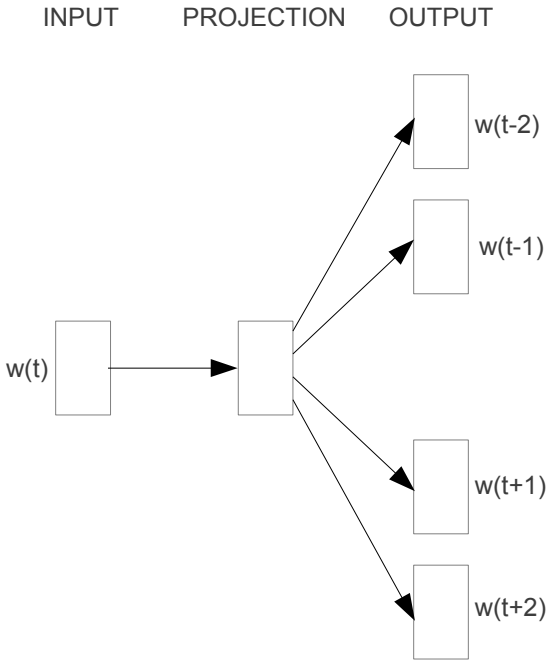
The word vectors are initialized randomly, the hidden layer is initialized with zeros. For each word in a context window, the surrounding word vectors are firstly summed (averaged) then, the averaged vector is projected by Hierarchical Softmax to the central word. Navigating through the path updates all involved nodes in the hidden status, then the error is back-propagated back to the input context word vectors separately.

Figure 4–3 **Continuous Bag of Words**

Iacobacci,, Pilehvar and Navigli introduced a lexical-semantic knowledge based sense embedding system in [12]. The sense inventory base is BabelNet [30], and the labeling algorithm is Babelfy [13]. This sense tagging algorithm makes use of a graph structure generated from BabelNet: the input text helps generating a subgraph of the network, then a dense subgraph heuristic is used to navigate through the subgraph to search for a semantic value for each content word.

Iacobacci et al. use this method to generate a sense-annotated corpus, then send the disambiguated corpus to word2vec toolkit, using CBOW architecture to obtain sense vectors.

Fadaee, Bisazza and Monz propose using *topic* instead of *sense* to induct sub-word semantic unit into the embedding system [9], as Yao and van Durme assume in [31] that *topic* and *sense* are interchangeable. The basic context unit in this article is a document. The nonparametric



The word vectors are initialized randomly, the hidden layer is initialized with zeros. Given one word surrounded by other context words, for each contextual word, the central word is projected by hierarchical softmax onto vocabulary space with internal nodes. The hidden layer is updated with the activation of each context word with each internal node in the binary tree, error is back-propagated to the central word vector.

Figure 4–4 **Skip-Gram**

Bayesian model used by Fadaee et al. is Hierarchical Dirichlet Process. HDP is used to determine the topic of a given document. There are three different topic-sensitive representation models proposed in the article: a representation for the couple of the word and the one selected topic (HTLE) , a representation for the generic word and the one topic selected separately (HTLEadd) and thirdly a representation for a probabilistic distribution over a set of topic-word pairs (STLE) (see Fig 4–5).

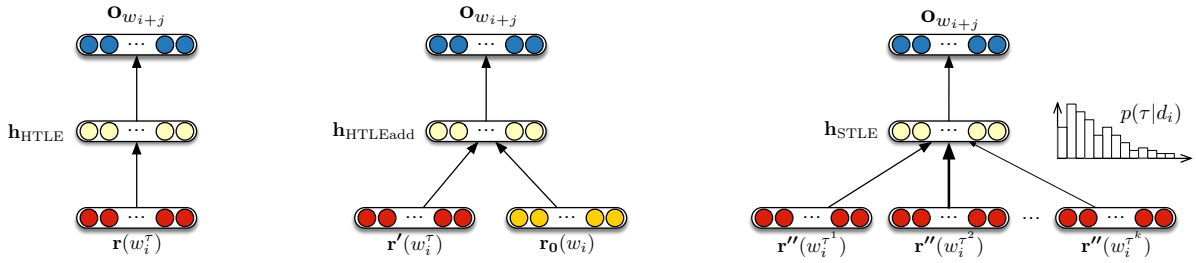


Figure 4–5 **Three Topic-Sensitive Representation Models**

The training concept to obtain embeddings is very similar to Skip-Gram model. With an input word, the training objective is to maximize the log-likelihood of context words. For HTLE and HTLEadd representations, the likelihood is written in Eq 4–2. Context word representations use unlabeled words to reduce the potential sparsity problem. For STLE representations, the likelihood is Eq 4–3, where the representation of w_i is independent from $topic$.

$$\mathbb{L} = \frac{1}{N} \times \sum_{i=1}^{N} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log \mathbb{P}(w_{i+j} \| w_i^{topic}) \tag{4–2}$$

$$\mathbb{L} = \frac{1}{N} \times \sum_{i=1}^{N} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log \mathbb{P}(w_{i+j} \| w_i, topic) \tag{4–3}$$

When updating the sense representations, different mechanisms are applied. For HTLE, the topic being learned from HDP, a sense of a word is directly retrieved from a vector table, see Eq 4–4. For HTLEadd, the sense is an addition of the basic word vector and the topic variation of the word, see Eq 4–5. For STLE, the input sense is a probability-weighted sum of

word-topic vectors, see Eq 4–6. w_i is a word in the vocabulary, r is the lookup table for raw word vectors, r_0 is for word-topic vectors, r_1 is for the variation vectors brought by a particular topic attributed to one word, r_2 is similar with r_0 except the vectors inside was generated for a probability distribution for topics.

$$h_{HTLE}(w_i) = r_0(w_i^{topic}) \tag{4–4}$$

$$h_{HTLEadd}(w_i) = r_1(w_i^{topic}) + r(w_i) \tag{4–5}$$

$$h_{STLE}(w_i) = \sum_{j=1}^{T} \mathbb{P}(topic_j \| document_i) \times r_2(w_i^{topic_j}) \tag{4–6}$$

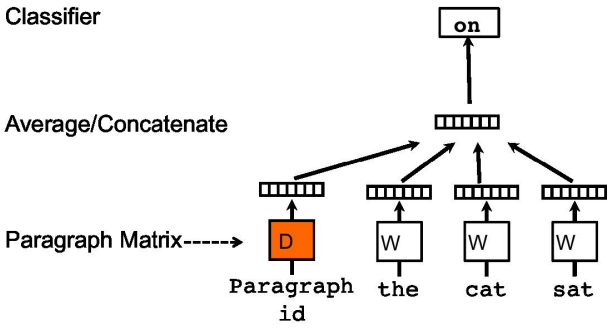
This implementation is borrowed to our model: a clique is like a topic attributed to one word, we could use similar approaches to obtain sense vectors.

4.1.3 Document Embedding

Also based on word embedding, the Document Embedding proposed by Le and Mikolov [32] enables a length-fixed vector to represent variable length pieces of texts. This training gives a better performance on text representations than simplistic bag-of-words and bag-of-n-gram models.

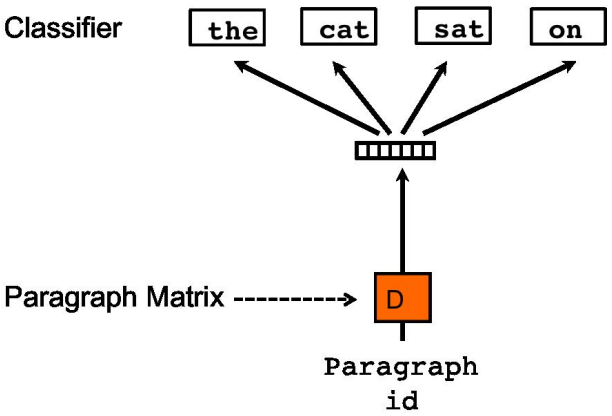
The article introduces two different models for generating *Paragraph Vector*s: the first one being Distributed Memory Model of Paragraph Vectors (PV-DM), which memorizes the word order, and given previous words, the model is able to predict the next word in the sentence (see Fig 4–6); the second one being Distributed Bag of Words (PV-DBOW), as it does not conserve the information about word order (see Fig 4–7).

After training, the vectors obtained are proven to be able to perform sentiment analysis, information retrieval tasks, which is why we decide to also profit from its performance to build clique vectors with the PV-DBOW model using clique elements.



With this model, we could either concatenate or average the vectors of previous words, the paragraph vector acts as a memory of the missing information of the paragraph. The number of previous words is determined as a parameter, however using the technique of sliding window the model is able to generate a whole paragraph (or document).

Figure 4–6 **Distributed Memory Model**



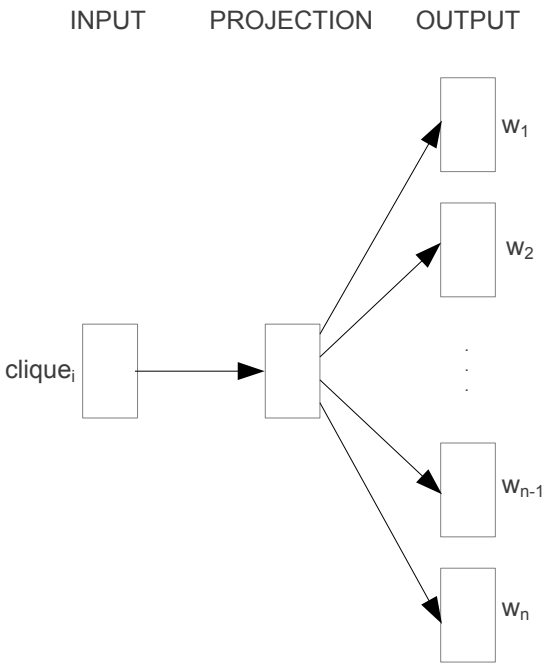
Very similar to Skip-Gram model, this model trains the vector to predict words in a small context window, except that word order is not a concern.

Figure 4–7 **Distributed Bag of Words Model**

4.2 Use clique words to train clique vectors

A training process avoiding annotating the whole corpus is developed: we train the clique vectors as documents using the clique words. We adopt Document Embedding's concepts to develop vector representation of cliques, since each clique is composed of contexonyms, this is based on the Latent Semantic Analysis assumption that words of similar meanings will occur in similar contexts (first introduced in Section 1.1), if a clique is composed of similar contexonyms, they tend to have similar meanings.

Based on Skip-Gram (which is also the prototype of PV-DBOW model introduced in Section 4.1.3), we implement the training model named *clique2vec* in python (see Fig 4–8), and we used Hierarchical Softmax to accelerate our training process. We supplied the system with randomly activated clique members in each iteration, and the clique occurrence is also shuffled.



The clique words are activated randomly each time. Clique word order is not a concern in this model as word order is also shuffled each time.

Figure 4–8 **clique2vec Clique Vector Training with clique words**

To set the training up, we use averaged clique word vectors trained on Mikolov's word2vec

toolkit to initialize our clique vectors to pursue a quicker eventual convergence.

In Table 4–1, we show a set of examples of top 10 similar cliques trained on the clique set extracted from English Wikipedia with 20 iterations, during the process of which we do not apply subsampling techniques.

With tests, the clique vectors (as the result of the training) are not in the same space with word vectors, as during the training we kept the word embedding unchanged. The vectors in clique embedding space is totally unrelavent to similar vectors in word space.

While the training is successful since the cosine distance of cliques correspond perfectly to the differences of clique contexonym sets, we observe that the vectors obtained by this training is not sufficient nor appropriate to establish a semantic embedding: clique contexonym information is only able to reproduce words (cliques) belonging to same context, but the cliques or the words associated are not similar with each other. See $good_0$ is associate words such as *reputation*, *luck*, *situation*, *guy*, whereas $good_{150}$ is more quality describing given *big*, *whole*, *clear*, *decent*. But *good*, *guy* and *whole* are hardly considered as similar words. The vectors is tested to memorize context information, but context information is too general to retrieve semantic information of one single lemma.

This suggests that cliques, although defined by its member elements, are not strong enough to determine its proper sense. The embedding should consider the headword and the clique couple as an atomic semantic entity.

We notice that the Python library gensim[1] provides a doc2vec model which is an optimized Python implementation of Le Quoc's [32] Paragraph Vector model. We use the framework to recheck our conclusion above.

This time we trained the word vectors and clique vectors altogether in a lower dimension (set to 200), expecting to see if the word vectors and document (clique) vectors belong to the same dimension. In Table 4–2, we listed the similar cliques (and words as this time they are trained together) of same cliques as a comparison with our clique2vec model. With the words and cliques trained together, the semantic regularities of entities still could not be established with vector distances.

[1] `https://radimrehurek.com/gensim/index.html`

Table 4–1 **Similar Cliques Trained on our clique2vec Implementation**

Clique	Similar Cliques	Cosine distance	Clique Contexonyms
$good_0$[a]			behaviour, offender, bad
	$reputation_7$	0.984849	behaviour, offender, bad
	$good_1$	0.976194	behaviour, habit, bad
	$reputation_6$	0.974196	behaviour, habit, bad
	guy_{13}	0.948492	behaviour, offender, bad
	$luck_6$	0.943307	behaviour, offender, bad
	$luck_{131}$	0.940681	offender, check, bad
	guy_{180}	0.940215	offender, check, bad
	lot_{435}	0.934790	recession, bad
	$luck_{125}$	0.934303	notoriously, awful, bad
	guy_{285}	0.933847	mistake, check, bad
$good_{150}$			perfect, brilliant, it's, beautiful, pretty
	big_{180}	0.979464	perfect, brilliant, it's, beautiful, pretty
	$good_{445}$	0.968112	brilliant, it's, amazing, exciting, beautiful, pretty
	big_{257}[b]	0.961148	brilliant, it's, clever, smart, beautiful, pretty
	big_{268}	0.960138	brilliant, it's, amazing, exciting, beautiful, pretty
	$whole_{292}$	0.957043	it's, nice, wonderful, smart, beautiful, pretty
	$sort_{733}$	0.953914	it's, nice, wonderful, smart, beautiful, pretty
	big_{311}	0.951851	it's, clever, pleasant, nice, beautiful, pretty
	$little_{21}$	0.946727	sad, it's, everybody, really, pretty
	$clear_{329}$	0.946449	entertaining, brilliant, it's, clever, wonderful, funny, charming, pretty
	$decent_{172}$	0.945818	scary, you're, it's, pretty, weird, stuff, kinda, that's
	$little_{33}$	0.945504	sure, it's, everybody, pretty
	$decent_{168}$	0.944113	scary, it's, pretty, funny, he's, stuff, that's, It's
	$sure_{101}$	0.943185	silly, sad, it's, funny, charming, pretty

[a] The subscript is the clique number with a numerotation along the clique report order at generation time.

[b] Some cliques of one same word with close clique numbers and similar contexonyms are omitted.

The vector dimension is 500. We see that the vector training is very successful as similar vectors do correspond to very similar (some exact) contexonym inputs.

Table 4–2 **Similar Cliques Trained on gensim Doc2Vec**

Clique	Similar Cliques	Cosine distance	Clique Contexonyms
$good_0$			behaviour, offender, bad
	guy_{13}	0.966367	behaviour, offender, bad
	$luck_6$	0.966150	behaviour, offender, bad
	$therefore_{427}$[1]	0.962055	
	$reputation_7$	0.961240	behaviour, offender, bad
	guy_{182}	0.915524	tough, offender, bad
	$reputation_6$	0.906363	behaviour, habit, bad
	$good_1$	0.899556	behaviour, habit, bad
	$luck_4$	0.882839	behaviour, outcome, situation, condition, good, bad
	$luck_8$	0.882157	behaviour, outcome, situation, good, bad, wrong
	$situation_{108}$[2]	0.880549	temper, habit, bad
	$continually$	0.768750	
	$worse$	0.765768	
	$keenly$	0.731858	
$good_{150}$			perfect, brilliant, it's, beautiful, pretty
	big_{180}	0.977940	perfect, brilliant, it's, beautiful, pretty
	$happily^*_{156}$[3]	0.975206	perfect, brilliant, it's, ever, guy, amazing, truly, funny, beautiful
	$great^*_{323}$	0.974943	perfect, brilliant, it's, beautiful
	$since^*_{93}$	0.972106	perfect, brilliant, it's, beautiful
	$ready^*_{218}$	0.970728	perfect, perfectly, brilliant, it's, wonderful, beautiful, pretty
	$happily_{249}$	0.965866	perfect, thing, it's, ever, guy, amazing, truly, funny, beautiful

Table 4–2 – *Continued on next page*

[1] *therefore* is not listed as most similar words in clique2vec.
[2] *situation* appeared as top similar words except that it is not among the top 10.
[3] Words with superscript * are new words compared to clique2vec model.

Table 4–2 – *Continued from previous page*

Clique	Similar Cliques	Cosine distance	Clique Contexonyms
	$happily_{163}$	0.961146	perfect, brilliant, ever, remarkable, memorable, amazing, truly, beautiful
	$happily_{155}$	0.960652	perfect, brilliant, anything, it's, ever, guy, certainly, truly, funny, beautiful
	$happily_{159}$	0.960545	perfect, brilliant, anything, ever, guy, certainly, great, truly, beautiful
	$none^{*}_{418}$	0.960283	it's, brilliant, beautiful, perfect, perfectly
	$compliment$	0.897860	
	$tonight$	0.870951	
	rid	0.835316	
	$marvellous$	0.833545	
	go	0.829029	

The vector dimension is 200. The similar words in terms of cosine distance are also added.

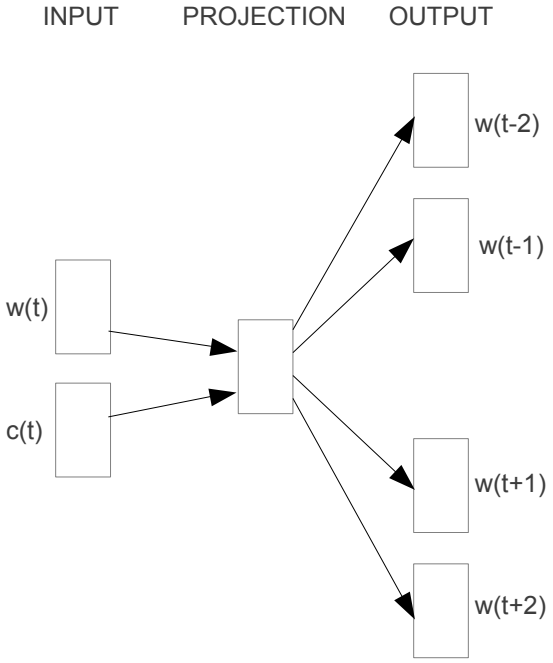
From comparison of the two embeddings, the similarity ranking for $good_0$ is relatively unchanged compared to clique2vec model, but $good_{150}$ has a pretty variant result.

4.3 Use sense-annotated corpus to train clique vectors

In this section, we introduce the method of obtaining clique embedding with a modified word2vec process, in which the leared corpus is annotated, and we add the training of cliques on top of the training of words (very similar to HTLE introduced in [9]).

An advantage of this embedding is that we do not construct vectors for clique that does not occur or rarely occur in real-life contexts.

Shown in Fig 4–9, we modified the Skip-Gram model to accomodate the notion of clique. Structurally, the clique is treated as a special kind of word. Each time a central word's vector is updated, the corresponding clique's vector is also updated.



The contexual information is given by words, but the center word is updated twice during training: one for the word vector, one for the clique the word is attributed with. Since differently labeled words activate different cliques, but they could activate the same word clique, clique vectors are less frequently updated than word vectors.

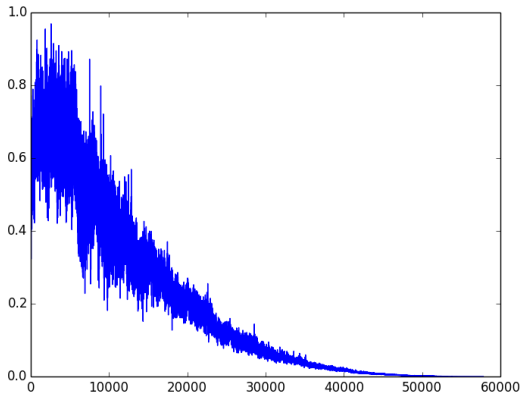
Figure 4–9 **HTLE clique2vec Double Vector Training with annotated corpus**

This training scheme (HTLE clique2vec) provides several advantage that the clique2vec model over clique contexonyms could not:

- It allows the words from which we could not extract cliques to be embedded.
- A portion of cliques would rarely occurring in the corpus, the embeddings for those cliques are not generated.
- This model trains words altogether with cliques, so different level of semantic information could be obtained.
- This embedding is more general in terms of construction, it conserves more properties of the classique word2vec embedding.
- Provided similar annotated corpora, more geometric properties could be exploited. In Chapiter VI we will be giving an example.

During the training with the sense annotated lemmatized wikipedia, each context lemma is activated when a lemma / clique in its context window is accessed. Through activation of projection layer, the lamma and the clique obtains a probability distribution over the vocabulary. For each context lemma, the probability distribution is matched with Hierarchical Softmax, the error is backpropagated.

This double stack training modifies the projection layer and updates the lemma / clique vectors. The process follows stochastic gradient descent (see Fig 4–10), so that finally with convergency, the projection layer could hopefully project the lemma / clique vectors to a probability distribution over its closely related context lemmas.



The time-window average of norms of each Stochastic Grandient Descent update follows roughly an inversed parabola since the alpha is linearly descending throughout the training.

Figure 4–10 **Stochastic Gradient Descent Updates Evolution**

The cliques are used for annotation, representing a contextual semantic variation. This enables the training process to train different senses separately, as only in particular context would the clique be updated, instead of making all the updates on a unified vector, which provides a vector in the midway.

Chapter V

Experiments and Evaluations

5.1 Generating contexonym cliques

We performed clique extraction on English/French Wikipedia dumps[1] and Gigaword Second Editions[2].

5.1.1 Statistics

Our development data for construction the cliques was extracted from the French Wikipedia. French Wikipedia has 372 315 608 tokens, 2 476 792 unique tokens. After applying French TreeTagger to the corpus, we have 245 877 754 lemmas, among which exist 1 714 307 unique nouns, verbs, adjectives and adverbs. Fig 5–1 gives the histogram of the lemmas.

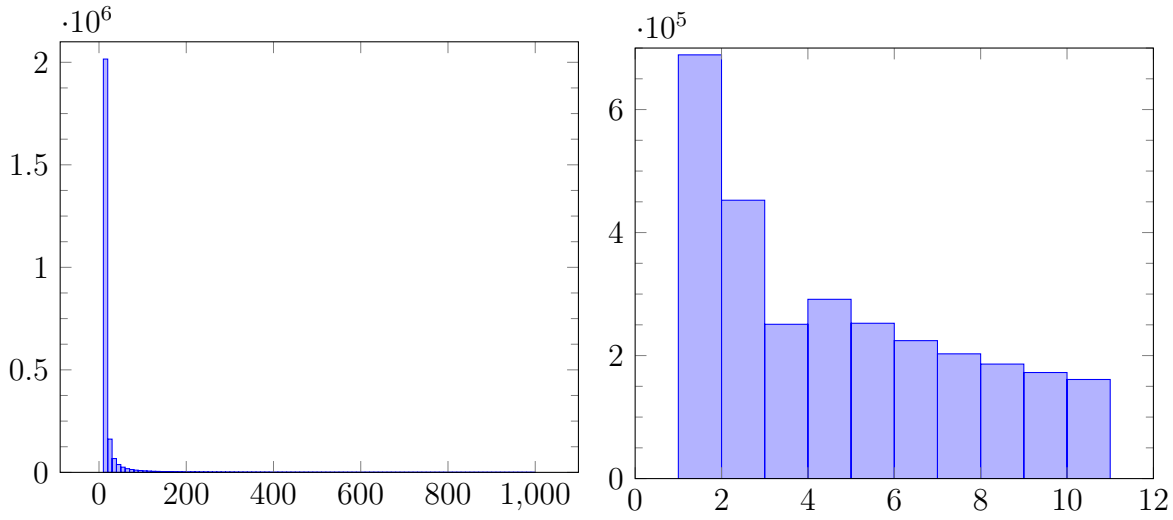
The Gigaword corpus is used for comparison, to validate the pertinence of clique contexonyms and the target word.

The parametered configured on French Wikipedia is then applied to English Wikipedia for clique extraction, then the English cliques are used to build embeddings and perform evaluation tasks. The English Wikipedia has 5 299 499 unique tokens, which is two times the scale of French wikipedia.

Under the assumption that less frequent words have fewer probability of being polysemous, which is introduced and discussed in [33], we are able to not take the words of least frequency into consideration, and it would save a lot of computational resource and improve the overall performance as the removal is about the largest histogram groups of lemmas.

The most frequent tokens occurring in French Wikipedia (Example 5.1) are mostly composed by functional words (some of which are homonyms). They are removed by the process

[1] `https://dumps.wikimedia.org/`
[2] `https://catalog.ldc.upenn.edu/docs/LDC2009T28/0readme.txt`



(a) Raw (b) Lemmatized

Typically, the histogram is following an exponential distribution.

Figure 5–1 **Histogram of (Un)Lemmatized French Wikipedia**

of POS tagging performed by TreeTagger.

Example 5.1 (Selection of Most Frequent Words Extracted from French Wikipedia)**.** *en*, *sa*, *s*, *deux*, *comme*, *été* (past tense for *be*, or *summer*), *et*, *avec*, *une*, *sont*, *ce*, *d*, *est* (third person singular for *be* or *East*), *l*, *pas*, *a*, *ou*, *les*, *qui*, *son*, *à*, *la*, *pour*, *le*, *qu*, *elle*, *plus*, *mais*, *cette*, *dans*, *du*, *se*, *il*, *de*, *des*, *ses*, *par*, *un*, *sur*, *aux*, *au*, *que*

With inspection to the French Wikipedia, the vocabulary for which we perform later steps is reduced to around 60 000 entries. We apply this limit also to English Wikipedia, despite it having more unique tokens. To verify the frequency-polysemy assumption, we sampled a few filtered most frequent lemmas, and display them in Example 5.2. These words (except *intrude*) are non-polysemous, non-homonymous and non-heteronymous, some are even not "anglophonized", which justifies the removal of unfrequent lemmas from the English Wikipedia vocabulary.

Example 5.2 (Sample of Most Frequent Filtered Lemmas Extracted from English Wikipedia ($fr(w_i) \approx 630$))**.** *scullin*, *slaton*, *intrude*, *mtm*, *kitana*, *foz*, *bhabha*, *knelt*, *neda*, *tauern*, *stx*, *ación*, *tehachapi*, *indiamen*, *chennault*, *rifkin*, *goodridge*, *longacre*, *bristly*, *elas*, *satyanarayana*,

ranchos, *novum*, *īyeh*, *outrages*, *goodluck*, *corfe*, *baidu*, *pieridae*, *gavel*, *ramage*, *wallpapers*, *nonconformists*, *kak*, *diametrically*, *bunin*, *btecs*, *orde*, *chō*, *scribal*, *feelgood*, *batmobile*, *dampened*, *borbón*, *learjet*, *hongwu*, *reticulata*, *bitstream*, *kapital*, *maz*, *fishkill*, *frew*, *halladay*, *dadu*, *glasshouse*, *unopened*, *kurri*, *dimming*, *carotenoids*, *mayans*

The co-occurrence count is then obtained on a sentential level. The data structure used to store the co-occurrence information is reduced to 8 Gigabytes in memory with all the counters being 32-bit unsigned integers, which could be accommodated by consumer level computers. The inter-connection of the lemmas is dense: the lemmas in French Wikipedia have in average co-occurred with 11870 other lemmas, English version being 10129.

The co-occurrence counting information then is transformed into a PMI table by the formula introduced in Section 2.3.3. α is the lower limit of pair co-occurrence / lemma occurrence ratio, β being the upper bound. γ and δ are the percentages of co-occurring contexonym selected into the association table. ϵ is the PMI threshold conserving stronger edges in the clique subgraph. We set $\alpha = 10^{-3}$ and $\beta = 0.33$, $\gamma = \delta = 0.05$ and $\epsilon = 10^{-6}$ as default value. The parameters are preliminarily set to a larger value and the dynamic adjustment will shrink oversized or overly dense subgraph to an appropriate status, meanwhile we could still find some information for more loosely connected lemmas.

5.1.2 Clique Enumeration and labeling

The same clique enumeration process (as introduced in Section 2.4.2) is applied to French Wikipedia, French Gigaword, English Wikipedia. The clique enumerations for around 60 000 words finished in less than 6 hours on the test machine using one process coded in Python.

A comparison between cliques extracted from Wikipedia and Gigaword is made. We can see that cliques from Wikipedia has a wider coverage of topics.

To examine the clique quality, we labeled sentences extracted from the United Nations Parallel Corpus (French). Here is an example in Example 5.3.

Example 5.3 (Sentence labeling Test)**.** C'est pourquoi nous nous attelons à la tâche difficile qui consiste à mettre en lumière et à présenter à la communauté internationale les actes de violence

perpétrés contre notre peuple, actes que l'on a occultés pour pouvoir les poursuivre en toute impunité et prévenir les sentiments de honte, d'indignation et de révolte qu'ils feraient naître dans la conscience morale de l'humanité.

(Google Translation: That is why we are addressing the difficult task of highlighting and presenting to the international community the acts of violence perpetrated against our people, which have been overshadowed in order to prosecute them with impunity and to prevent Feelings of shame, indignation, and revolt, which they would produce in the moral conscience of humanity.)

We can see that the clique conteonyms are activated by the lemmas in the long sentence, which in return gives cliques of more related contexonyms. The words correspond well to the topic of the sentence.

Table 5–1 **A Clique-Labeled Sentence**

word	POS	lemma	clique contexonyms
[···]			
pourquoi	ADV	pourquoi	chose comment répondre n'est clairement connaissance pourquoi que rien nécessité savoir avis fait vérité question vraiment qu'elle justifier c'est qu'il qu'ils montrer qu'on n'a expliquer si on
[···]			
tâche	NOM	tâche	direction confier soin mission gestion responsabilité architecte conception tâche réalisation
difficile	ADJ	difficile	possible nécessiter faible utile difficile relativement condition voire efficace dur sensible très extrêmement particulièrement lent facile avérer rare rarement
[···]			
peuple	NOM	peuple	délégation unir paix sécurité résolution uni souveraineté maintien peuple africain charte organisation
,	PUN	,	
[···]			
de	PRP	de	

Table 5–1 – *Continued on next page*

Table 5–1 – *Continued from previous page*

word	POS	lemma	clique contexonyms
honte	NOM	honte	prévenu innocence acquittement accusé verdict atténuant culpabilité présomption
,	PUN	,	
d’	PRP	de	
indignation	NOM	indignation	susciter soulever inquiétude vif
[···]			
humanité	NOM	humanité	coupable perpétuité crime aggravant pénal passible délit viol sodomie punir commettre
.	SENT	.	

5.2 Clique Embedding

We build three clique embeddings with the two models described in Section 4.2 and 4.3.

We pre-trained a CBOW model on lemmatized English Wikipedia, the dimension is set to 500, context window is set to 8, subsampling is set to 10^{-4}, and the program run through the corpus with 15 iterations.

For our Sense (clique) Embedding, firstly, in the clique2vec method where we initialized the clique vectors using the average of clique contexonym vectors, we build an embedding with the clique lemmas with 20 iterations through the clique set, the maximal number of activated contexonym is set to 8. The dimension of vectors is 500, the same as the pre-trained word vectors. We trained the vectors firstly with orderly iterations, then we changed the model to uniformly randomly retrieve a clique from the set while maintaining the property that each clique is input at least 20 times. The example is shown before in Example 4–1.

We also tested the doc2vec model with gensim library. The Skip-Gram window is configured to 8, the subsampling ratio is 10^{-5}, and the cliques are iterated 20 times. We trained a set of

200-dimension vectors and a set of 400-dimension ones. To check the training result examples, refer to Example 4–2.

With limited experiment time, we labeled 786 508 sentences, 5 966 697 lemmas from the lemmatized English Wikipedia. The sentences are uniformly distributed over the English Wikipedia dump. We build a traditional word2vec Skip-Gram model on this small corpus and a HTLE clique2vec double training Skip-Gram model. The window size is set to 8, the subsampling ratio is 10^{-4}, the vectors are obtained with 200 iterations (as default iteration time is set to 5) due to the lack of annotated data.

In this test set, we have trained 60825 unique cliques (after rejecting clique with occurrence less than 5), belonging to 6631 distinct lemmas. If the less frequent cliques are conserved, we have 222 858 cliques belonging to 139 914 distinct lemmas.

The similarity result for raw word2vec Skip-Gram training is not promising due to the sparsity and the lack of corpus, and many cliques are missing from the partially tagged corpus. Here we give the most relevant cliques in the built embedding for $bright_2$ and in Table 5–2.

Whereas with the adopted HTLE clique2vec model, the result is slightly better than the raw version. With shown example which is presented in Table 5–3, we see that HTLE clique2vec model at least gives relevent semantic entries with proper distance, rather than occurring words.

We do not perform further systemized word similarity benchmark, as the vocabulary does not cover enough frequent words. This drawback is also shown in later Lexical Substitution Task results.

5.3 Lexical Substitution Task

5.3.1 Introduction

The English Lexical Substitution Task (LST) [34] is proposed for SemEval[1], the series of computational semantic analysis system evaluations. In this task, the candidate language systems have the objective to find one or several appropriate substitutes for a target word in a given context. Since only a short context is given and no other reference is provided, it is up to the system to disambiguate the text and use its proper linguistic inventory to propose substitution

[1] `http://nlp.cs.swarthmore.edu/semeval/`

Table 5–2 **Similar Cliques/Lemmas**

Clique	Similar Clique/Lemma	Cos distance	Clique Contexonyms
$bright_2$			insolation, cloudiness, cloudless, sunshine
	$light-year_{20}$	0.595697	confirmed, habitable, extrasolar, exoplanets, exoplanet, parsec, main-sequence, Sun-like, super-Earth
	$star$[a]	0.590669	
	sky_{213}	0.560191	celestial, solstice, moon, ecliptic, zodiac, sun, sunrise, lunar, rising
	$faint$	0.541700	
	$constellation_0$	0.529212	Australis, constellation, asterism
	$Ophiuchus$	0.519178	
	$yellow_{24}$	0.506382	apricot, pineapple, orange, citrus, mandarin, tangerine, avocado, lemon, guava, grapefruit, papaya, juice
	$galaxy_{86}$	0.500782	leathery, elliptical, ovate, acuminate, petiolate, oblanceolate, obovate
	$orange_{29}$	0.489912	alder, ponderosa, lodgepole, grove, old-growth, birch, poplar, cypress, cottonwood, redwood
	$dwarf$	0.489621	
	$supergiant_{115}$	0.484539	glaucous, needle-like, stoma, stomatal, blue-white
	$light-year_0$	0.483853	stoma, stomatal, blue-white, needle-like, glaucous
	$dark$	0.479842	
	$parsec_{14}$	0.469912	confirmed, habitable, extrasolar, exoplanet, exoplanets, parsec, main-sequence, Sun-like, super-Earth
	$bright_6$	0.466070	convective, cloudiness, overcast, sunshine
	$magnitude_{215}$	0.463203	earldom, holder, apparent, baronetcy, presumptive
	$blue-white_3$	0.462852	light-year, extrasolar, main-sequence, exoplanet, Sun-like, super-Earth, parsec
	$planet$	0.456509	
	$main-sequence_6$	0.438156	white-fronted, wigeon, black-necked, swan, Cygnus, green-winged, pochard

[a] Words without subscripts are words without cliques. They are regarded as the unique clique of itself during training.

The vector dimension is 200. The similarity is in general drastically dropped compared to clique2vec and doc2vec methods. Furthermore, the similarity set is much more heterogeneous as there are rarely multiple cliques of one same lemma.

Table 5–3 **Similar Cliques/Lemmas**

Clique	Similar Clique/Lemma	Cosine distance	Clique Contexonyms
$bright_2$			insolation, cloudiness, cloudless, sunshine
	$bright$[a]	0.981421	
	$bright_6$	0.745300	convective, cloudiness, overcast, sunshine
	$bright_9$	0.631594	insolation, sunny, overcast, sunshine
	$sterne$	0.624706	
	$fourth - brightest$	0.622323	
	$fourteenth - brightest$	0.604923	
	$sixth - magnitude$	0.590672	
	$aril_{37}$	0.585636	protuberance, papilla, parapodium, fleshy
	$fifth - brightest$	0.585571	
	$thirty - first - brightest$	0.582371	
	$light$	0.572362	
	$light_{24}$	0.568542	light, darkness, shadow, eclipse, brightness, daylight, sunlight

[a] Words without subscripts are words without cliques. They are regarded as the unique clique of itself during training.

The vector dimension is 200. Words and cliques are co-trained together. The raw distance table presented much more noise as the 200 iteration has magnified local disturbances. The embedding is able to establish the similarity between $bright$, $magnitude$ and $light$ with only 6 million tokens in the training corpus (usually Skip-Gram training requires more than 1 billion tokens).

candidates.

This task propose three subtasks: the first one being **best**, where the system evaluated could provide multiple guesses, but the credit attributed to fitting answers are divided by the number of guesses, the second one being **oot**, where the system could offer up to 10 candidates, the third one being **mw**, where the system is to determine if the target is intergrated into a multiword phrase. With our proposed models and our clique labeling system, we will be performing the **oot** task as the vector similarity catches the relevence, antonyms are also considered similar in word embeddings, and our data source till the day of drafting of the thesis is still not big enough to produce accurate results.

In the given task, the target words for which the candidate substitutions are for are also supplied with Part-Of-Speech (POS) information, which our clique model does not take into account.

Example 5.4 (Extract from LST Test File)**.**

```
<?xml version="1.0" ?>
<!DOCTYPE corpus SYSTEM "lexsub.dtd">
<corpus lang="english">
<lexelt item="side.n">
        <instance id="301">
            <context>On Sunday at Craven Cottage , Jose Mourinho and his all
        stars exhibited all of the above symptoms and they were made to pay the
        price by a Fulham <head>side</head> that had in previous weeks woken up
        after matches with their heads kicked in .</context>
        </instance>
</lexelt>
</corpus>
```

5.3.2 System Explained

To complete the task, we first label the target word in the sentence with our labeling technique. The sentence is firstly lemmatized. For example, the sentence in Example 5.4 is lemmatized:

Then the lemmas inside the sentence are filtered according to POS tags and vocabulary coverage. The sentence is therefore reduced to: **star all above symptom price side previous**

week match head.

For the contexonyms of *side*, we matched them with the context lemmas. We observe that even after the activation of similar contexonyms, the contexonyms are still not activated with the word embedding cosine similarity threshold set to 0.7. In this case, the system could either return "NIL" as a response as it cannot distinguish which sense the context indicates, either return the most common clique to guess rashly. For the displayed results, the system does not give a clique label for those words.

But for another example, such as *He responded , " Good God , girl , can't you understand English !"*, with the target word being **girl**, the phrase is reduced to **respond girl English**, which activates the contexonyms **son, brother, elder, eldest**. The calculation of χ^2 distance yields the 106th clique of *girl*: **girl, prince, uncle, son, young, brother, sister, mother, boy, daughter, elder, eldest**, which represents a family-related value rather than a professional-related one, such as *an office girl*.

As the target word in sentences given in the task are labeled with cliques, we use then our embedding system to propose alternative candidates. The metric is cosine distance, a notion adopted in word2vec toolkits. The cliques of one same word is merged as one entry, the similarity used is the similarity of the first occurrence. We set the distance threshold to 0.7, if there are more than needed proposals above the threshold, the tops are selected.

Word	On	Sunday	at	Craven	Cottage	,	Jose	Mourinho	and	his	all	stars
POS	IN	NP	IN	NP	NP	,	NP	NP	CC	PP$	DT	NNS
Lemma	on	Sunday	at	Craven	Cottage	,	Jose	Mourinho	and	his	all	star

exhibited	all	of	the	above	symptoms	and	they	were	made	to	pay	the	price
VVN	RB	IN	DT	JJ	NNS	CC	PP	VBD	VVN	TO	VV	DT	NN
exhibit	all	of	the	above	symptom	and	they	be	make	to	pay	the	price

by	a	Fulham	**side**	that	had	in	previous	weeks	woken	up	after	matches
IN	DT	NP	**NN**	WDT	VHD	IN	JJ	NNS	VVN	RP	IN	NNS
by	a	Fulham	**side**	that	have	in	previous	week	wake	up	after	match

with	their	heads	kicked	in	.
IN	PP$	NNS	VVN	IN	SENT
with	their	head	kick	in	.

5.3.3 Performance of Clique Labeling

Before obtaining the evaluation score of the task, it is interesting to examine if the Clique labeling process itself is effective enough to distinguish different context, especially with the introduction of contexonym expansion based on word vector similarity.

The LST task provides a baseline system, which selects synonyms from WordNet with the largest frequencies obtained from British Nation Corpus[1]. Due to the simplistic design of the baseline system, it is providing unique candidates to every word of a particular Part-of-Speech.

Whereas our system is providing multiple cliques. For the 279 instances provided in **trial** set, we are able to see that the clique labeling is able to provide appropriate clique selections based on the context, but it still lacks the precision of determining the word's proper sense. For around 10 instances of each target word, our labeling system gives in average more than 5 clique variations.

In Table 5–4, the labeling for the noun "examination" has distinguished the two different contexts regarding to the word itself, despite the difference of the context thus different clique choice, the lexical substitutions are quite similar.

5.3.4 Performance of Clique Embedding

Now we will profit from the labeled target words and the trained Clique Embeddings. The labels are replaced by candidates selected from different embedding models using cosine distance. The replacements are matched against the Golden standard given by the task, and a score is evaluated for each system. The LST result is presented in Table 5–5.

We tested 5 systems built using our provided data. The CBOW system is a classic CBOW model trained on the entire lemmatized English Wikipedia. The coverage is brought down on 89.7%, which is the upper limit of our other systems trained on the same data source.

The d2v model are trained as the clique member contexonyms are supplied as documents. If a clique is not provided by the labeling system, or the clique does not possess an embedding vector, similarities between words are referred to to provide candidates. This model works the

[1] `http://www.natcorp.ox.ac.uk/`

Table 5–4 **Clique labeling "examination"**

Sentence	Clique Contexonyms	Gold Standard[a]
However , the **examination** will be detailed if the applicant expressly requests it (for example , at the same time as filing the Demand) , or if the applicant files amendments and/or arguments (for example , in response to the International Search Report , with the Demand or in response to a Written Opinion ; see below) .	fee, criterion, competitive, university, exam, applicant, examination, undergraduate, admission, requirement	investigation, scrutiny, inspection
From the previous **examination** last week the auto-immune deficiency could also be excluded .	painstaking, thorough, methodical, meticulous	investigation, inspection, consultation, assesment
Readers will appreciate this well-researched **examination** .	painstaking, thorough, methodical, meticulous	investigation, scrutiny, enquiry, analysis
As well , he will continue his **examination** of the transatlantic connections that fuelled the slave trade .	painstaking, thorough, methodical, meticulous	investigation, scrutiny, study, inspection, analysis

[a] The Gold Standard candidates are mannually labeled by a group of British linguists, which is also the scoring standard.

Table 5–5 **oot results**

System	Precision	Recall	Mode P	Mode R	Coverage[f]
Baseline[a]	0.214	0.214	0.286	0.286	100%
CBOW[b]	0.157	0.141	0.250	0.228	89.7%
d2v[c]	0.060	0.048	0.095	0.078	79.7%
SG-S[d]	0.088	0.049	0.155	0.087	56.3%
SG-A[e]	0.092	0.032	**0.157**	0.053	35%
HTLEc2v	**0.148**	0.086	0.115	0.068	58.3%

[a] Baseline is based on WordNet and BNC synonymy frequency.
[b] CBOW word embedding trained on lemmatized Wikipedia.
[c] Document vectors trained on clique contexonyms. If the labeled clique does not have a trained vector, the lemma's vector is used for similarity selection.
[d] Skip-Gram applied against the partial unannotated corpus.
[e] Skip-Gram applied directly against sense-annotated corpus.
[f] The percentage of testing instances where the system provides candidates.

worst as the embedding learned by member contexonyms does not establish a semantic regularity as shown in Table 4–2.

The raw Skip-Gram over unannoteted and annotated partial corpus and HTLE clique2vec model suffer from the lack of clique-labeled data source. Trained on the same corpus, we can see clique labeling alone can provide a performance boost (SG-A over SG-S) though it may suffer from sparsity problem. HTLE clique2vec's double stack training scheme provides a better representation of words and cliques in the vector. The coverage is also ensured by clique to word fallback, and in this model, the words and the cliques belong to one shared space.

Furthermore, we can see a comparable result from our HTLEc2v model and CBOW model, whereas the former one trains only on a limited source.

Chapter VI

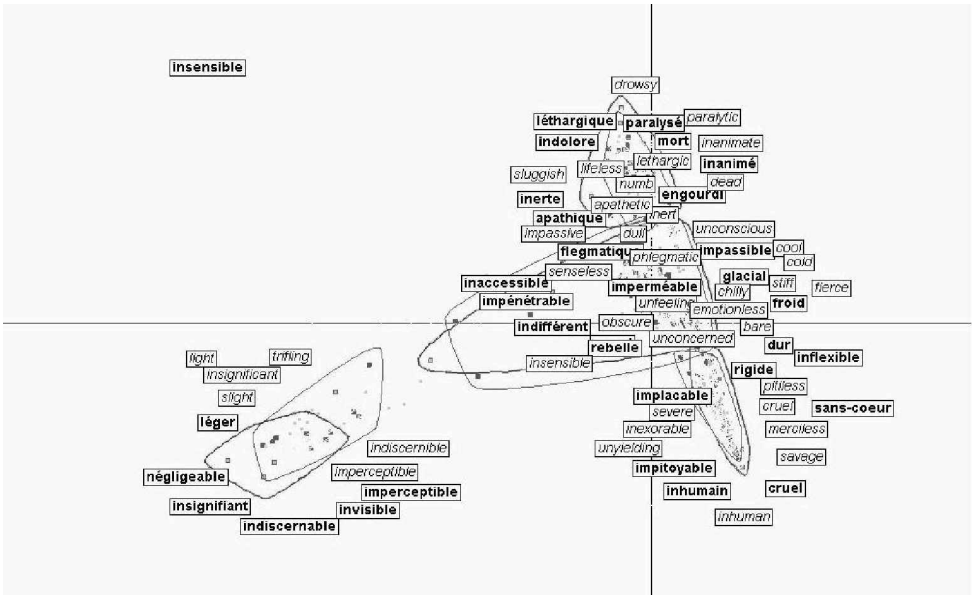
Application in Machine Translation

Several studies [35] suggest that a bilingual person has one common semantic neural system for two languages. Based on the found evidences, studies that try to associate semantic embeddings for different languages have achieved successful results, be it to reconstruct the common embedding for both languages.

In [18], Ploux and Ji developed a mapping strategy to construct a common synonym clique space for English and French synonymy cliques. The method is three-fold: firstly two separate clique spaces for source / target languages are constructed, each clique space has its own clique synonyms; secondly the usage of a translation dictionary helps to link the source synonyms and target synonyms, if multiple candidates for translation exist, a ranking system based on translation matrix is to select the most relevant translation; thirdly, the geometry manipulation to be done on target clique space is determined by factorial analysis, so that the two languages' cliques could be found in coherent regions. This mapping strategy is able to give pertinent clique positions. An example is shown in Fig 6–1, where the clusterings of cliques in both languages correspond well.

For Word Embeddings such as those trained by word2vec toolkits, Mikolov, Le and Sutskever find in [36] that word vectors does also present cross-language linear similarities once a PCA is done on the word embeddings. An example of English / Spanish embedding is presented by Fig 6–2, where the words in English Embedding and their translations in Spanish Embedding present similar spatial relational invariances.

With our HTLE clique2vec model, the clique vectors are believed to be able to separate different senses from one single polysemous / homonymous / heteronymous word, while maintaining the properties of word2vec trained vectors. If we apply the same clique enumeration process to corpora of similar structures in different languages, we expect that the labeling mech-



The four clusters of cliques for each language is pertinently mapped. The bold font boxes are French synonyms, the light font boxes are English Synonyms.

Figure 6–1 **French and English Clique Space for French "insensible"**

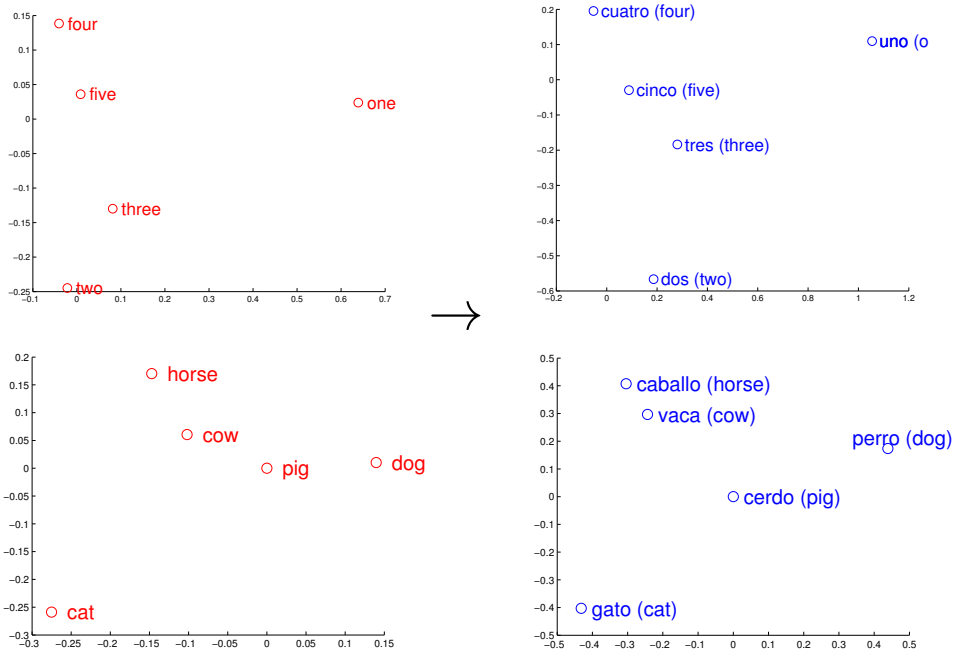
anism could separate the different contextual situations of the words in those languages, and the separated senses share one common language-independent semantic value.

To map the source and target language embeddings of the translation language pair, Mikolov et al. used s translation matrix to project one space onto another. The matrix parameters are learned from a dictionary generated from Google Translation results for the set of most frequent words in the source language corpus.

For our clique structures, the translation is based on clique-pair linkages, while the mapping between cliques is not necessarily a bijection, given that the clique granularity could vary depending on the corpus.

The linking passage between source language cliques and the target language clique could be done with Ploux et al.'s method in [18]. In dealing with the differences between synonymy cliques and contexonym cliques, an adaptation is to make to profit from contexonyms in different languages.

Since with synonymy cliques in different languages, even the sense items of one word can



The PCA projected spaces are rotated manually to more explicitly display the linear similarity. The semantic entities' arrangement suggests that a linear embedding could be found to map one space onto another, since PCA, rotation are all linear operations.

Figure 6–2 **English and Spanish Word Embedding Projections**

vary slightly, each sense item brings a word a set of synonyms, the linkage between the source word and the target word could still be established with one shared sense item. With contexonym cliques, we define with each clique a contextual environment, which links with the word its frequent noun entities, usual actions and common descriptions. While the contextual collocation can vary across languages, at least a minimal amount of collocations are invariant with proper translation techniques. The collocations are collected in one dictionary to be later referred to when generating relevant contexonyms.

To establish a mapping, the clique enumeration process is executed with forcefully added contexonyms. At the stage of clique contexonym selection to be later used to generate a subgraph, the contexonyms which have translation relations are given a higher priority then they have in usual contexonym selection processes. The conservation of the translation relations enables a future mapping linkage with the collocation dictionary. During the dynamic adjustment

of subgraph to accelerate the clique enumeration, these contexonyms are also conserved.

Should it be successful to establish a mapping with the added contexonyms, the clique clusters in both language are now found in coherent regions. To define clique-level translation, the pairing is set to be loose-fitting to solve the potential sparsity problem. The translation is valid for two cliques, if the cliques belong to the same cluster region. This information helps establishing the translation matrix introduced in [36] for HTLE clique2vec embeddings.

The cliques sharing similar senses usually have high cosine similarity, that is they are found in a common region in the embedding space. The evaluation of mapping precision uses conventionally the metric of cosine distance. Given a good quality of clique embedding, the similar cliques should be clustered in one same region.

A high density of similar vectors gives extra information on adjusting the translation matrix parameters, as pair-wise vector association information is abundant with similar clique vectors. The mapping could be tested to examine if a similar sentence-based word-level translation performance is offered by the model as the word embedding mapping reported in [36], which varies from 25% to 44% for one unique candidate proposal, 45% to 62% for five proposals.

Chapter VII

Follow-up Works

7.1 Involvement of Part-of-Speech Tags

The first manipulation applied on our clique extraction corpus is to lemmatize the corpus and reject words of unnecessary POS tags. For the consistency with previous work, in this thesis we do not add POS information to the clique as a preliminary element of clique extraction: we believe the contexonym clique itself will capture the grammatical component information.

A more stable way to conserve this information is to label the lemmas with POS tag, and generate different clique sets for each POS-lemma couple. This enables the semantic variation to be classified more clearly: syntactical values are separated from semantic values.

This could also give a performance boost to Lexical Substitution Task since the POS information supplied by the test set is now a considered factor, and actually used in the clique system.

7.2 Defragmentation of Contexonym Cliques

We believe the notion of clique is the minimal semantic unit, the continuous transition of cliques ensures a subtle semantic variation to be conserved. However it also brings the problem of fragmentation of senses, which could further worsen the lack of training data.

The clique system is typically providing more than 50 sense items for one word, whereas in WSD systems or dictionaries the number is usually limited to 10. It is envisaged that a method for merging similar cliques is to be applied to the clique set, to merge the sense items to provide a coarser, but yet minute sense representation model.

The Hierarchical Clustering technique used in [5, 18] is a possible solution. The cliques are merged given its close distance to other local cliques while maintaining a global classification. In merging the cliques, the list of contexonyms grow and the chance of the new clique being

activated by a contexonym is also increasing. The difference of χ^2 distances of new cliques are elevated compared with the former system, the selection among the clique-groups to decide which sense the labeled word possess would be less ambiguous.

If the clique numbers are reduced, the clique matrix shrinks. The calculation of the χ^2 distances between pseudo-cliques and cliques are also accelerated with fewer rows.

Another possible improvement is the precision of clique embedding. The coarser granularity of cliques merges the clique instances into one, providing more occurrences of single semantic entity. As the clique appears more frequently in the annotated corpus, the training updates the clique vector more often, providing a vector of more precision.

7.3 New Labeling Method

Currently, a clique is labeled to a lemma if the sentence-formed pseudo-clique has this clique as the closest one. To calculate the χ^2 distance it would require manipulations of large clique matrices, whereas the labeling process could be considered as an classification system where we input a series of contexonyms and get the output of a clique number. Actually, given a clique matrix, at each time the activation of a particular contexonym gives the almost the same influence on the distance if the matrix is sufficiently large. And if the matrix is small the choice of clique is more evident given the classification objective is more clearly defined.

That is to say, once we obtain a clique set for one headword, the training of a ML system (as simple as decision trees) could provide a more efficient labeling system than using permanently the direct calculation of distances.

Involving a lighter mechanism to label the corpus accelerates enormously the generation of clique annotated system. The accuracy of the mechanism would be even more improved if clique merging is also applied.

7.4 Soft-Annotated Corpus Training

In order to increase the occurrences of cliques, the soft labeling method could be adopted to annotate the corpus. Similar with the idea introduced in Section 4.1.2, each word is labeled

with a probability distribution over a set of cliques, which could be generated by Softmax. The soft labeling provides intermediate status of semantic value, even if the merging cliques give a coarser granularity, the soft label conserves the nuances.

Summary

This thesis proposes a combinatorial method of existing word embeddings and contexonym cliques to build semantic sense embeddings.

Using a graph based method to generate contexonym cliques in an unsupervised fashion enables the words in the context to be labeled with a semantic sense tag: contexonym clique. Then we use a Skip-Gram modified training scheme to obtain the embedding vectors for both cliques and words.

We first modify slightly the generation process of contexonym cliques: after obtaining the occurrence and co-occurrence statistics of the corpus, the multi-level filters and the dynamic adjustment to the clique generation subgraph for clique enumeration acceleration ensures a reasonable time and resource consumption to obtain the clique set for the concerned lexicon.

Secondly, we propose a sentence-based extensive contexonym activation mechanism to label the headword in the sentence with one of its cliques. A preliminary word embedding is used to calculate the similarities of the contexonyms. Similar contexonyms in the context activates the corresponding contexonym, which increases the chances of contexonym activation, partially solved the problem of sparsity. The labeling is a process of clique selection, based on the χ^2 distance of pseudo-clique formed using extensive contexonym activation.

With the labeling procedure we annotated a small corpus with text source extracted from English Wikipedia. We used HTLE clique2vec model to train a distributed vector embedding for cliques and words occurring in the corpus. We build other embedding systems based on clique member contexonyms using a document embedding techniques for exploration. Traditionally trained embeddings are also built with the same source for comparison.

The labeling system and the sense embedding are tested with English Lexical Substitution Task. With only a corpus ten times smaller compared to usual corpus used to obtain Skip-Gram embeddings, our system performs similarly with the word embedding system trained on the entire source corpus, outperforms other systems trained on the partial corpus, showing the

potential of our embedding model.

Appendix A

Contexonym cliques of *"good"* extracted from Wikipedia

1. Olympic, feat
2. Olympic, finish, marathon
3. Olympic, marathon, record
4. Olympic, wrestler
5. accident, bad, disaster, terrible
6. accident, bad, visibility
7. accident, badly, crash
8. accident, crash, deadly, disaster, fatality
9. accident, crash, deadly, disaster, terrible
10. accident, crash, disaster, fatality, tragedy
11. accident, crash, disaster, terrible, tragedy
12. accident, crash, drunk
13. accident, crash, visibility
14. acquaintance, admirer, friend, lifelong, lover
15. acquaintance, admirer, friend, long-time
16. acquaintance, admirer, friend, longtime
17. acquaintance, boyfriend, classmate, co-worker, crush, friend, girlfriend
18. acquaintance, boyfriend, classmate, co-worker, crush, friend, neighbor
19. acquaintance, boyfriend, classmate, crush, friend, friendship
20. acquaintance, boyfriend, co-worker, friend, neighbor, stranger
21. acquaintance, boyfriend, crush, friend, girlfriend, long-time
22. acquaintance, boyfriend, crush, friend, girlfriend, longtime
23. acquaintance, boyfriend, friend, girlfriend, lover

24. acquaintance, boyfriend, friend, lover, stranger
25. acquaintance, childhood, classmate, crush, friend, friendship
26. acquaintance, childhood, classmate, crush, friend, neighbor
27. acquaintance, childhood, classmate, friend, friendship, lifelong
28. acquaintance, childhood, narrator
29. acquaintance, circle, intimate
30. acquaintance, close, friend, friendship
31. acquaintance, close, friend, neighbour
32. acquaintance, close, relative
33. acquaintance, colleague, friend, lifelong
34. acquaintance, colleague, friend, long-time
35. acquaintance, colleague, friend, longtime
36. acquaintance, crush, friend, friendship, mutual
37. acquaintance, crush, friend, neighbour
38. acquaintance, friend, friendship, intimate, lifelong
39. acquaintance, friend, intimate, lifelong, lover
40. acquaintance, friend, intimate, lover, stranger
41. acquaintance, friend, neighbour, stranger
42. acquaintance, generous, stranger
43. acquaintance, girlfriend, lover, narrator
44. acquaintance, neighbor, relative
45. acquaintance, offender, stranger
46. admirer, affection, great
47. admirer, affection, jealous, lover
48. admirer, affection, lifelong, lover
49. admirer, biographer, lifelong
50. admirer, friend, jealous, lover
51. adventure, companion, imaginary
52. adventure, evil

53. adventure, teenage
54. affected, bad, disaster, drought, famine, flood, flooding
55. affected, bad, prognosis
56. affected, cure, prognosis
57. affected, cyclone, disaster, drought, flood, flooding
58. affected, cyclone, disaster, flood, flooding, tornado
59. affection, beautiful, beloved, jealous, lover
60. affection, beautiful, truly
61. affection, beloved, dear
62. affection, beloved, generous, loyal
63. affection, beloved, girlfriend, jealous, lover
64. affection, beloved, girlfriend, lover, narrator
65. affection, boyfriend, classmate, crush, friendship, jealous
66. affection, boyfriend, classmate, crush, girlfriend, jealous
67. affection, boyfriend, girlfriend, jealous, lover
68. affection, boyfriend, lover, stranger
69. affection, childhood, classmate, crush, friendship
70. affection, childhood, classmate, friendship, lifelong
71. affection, childhood, narrator
72. affection, crush, friendship, mutual
73. affection, dear, stranger
74. affection, dear, truly
75. affection, friendship, intimate, lifelong
76. affection, generous, loyal, stranger
77. affection, intimate, lifelong, lover
78. affection, intimate, lover, stranger
79. all-time, attendance, record
80. all-time, best-selling, selling
81. all-time, great

82. all-time, hat-trick, overtime, victory, win
83. all-time, hat-trick, try, victory, win
84. all-time, overtime, postseason, undefeated, victory, win
85. all-time, overtime, vacation
86. all-time, poll, postseason, undefeated
87. all-time, postseason, record, undefeated, win
88. ally, close
89. ally, defeat
90. ally, foe, long-time
91. ally, foe, longtime
92. ally, loyal
93. amazing, beautiful, brilliant, exciting, it's, pretty
94. amazing, beautiful, brilliant, exciting, it's, truly
95. amazing, beautiful, exciting, it's, pretty, thing
96. amazing, brilliant, feat, remarkable
97. amazing, brilliant, remarkable, truly
98. amazing, exciting, funny, it's, pretty, really, stuff, thing
99. amazing, funny, guy, it's, pretty, really, stuff, thing
100. amazing, narrator
101. ambitious, arguably, brilliant, exciting
102. ambitious, brilliant, exciting, it's
103. anybody, anyone, anything, anywhere, else
104. anybody, anyone, anything, else, hardly, it's
105. anybody, anyone, anything, ever, hardly
106. anybody, anything, else, hardly, it's, stuff
107. anybody, anything, else, it's, maybe, nobody, really, stuff
108. anybody, anything, else, it's, maybe, pretty, really, stuff
109. anybody, anything, ever, nobody
110. anybody, anything, guy, it's, maybe, nobody, really, stuff

111. anybody, anything, guy, it's, maybe, pretty, really, stuff, stupid
112. anybody, ass, guy, stupid
113. anybody, awful, it's, pretty, really, stuff, stupid
114. anyone, ever, unlikely
115. anything, before, ever
116. anything, certainly, hardly, it's, stuff
117. anything, certainly, it's, maybe, pretty, stuff
118. anything, certainly, it's, nothing
119. anything, else, it's, maybe, pretty, really, stuff, thing
120. anything, else, it's, nothing, really
121. anything, ever, funny, nobody
122. anything, funny, guy, it's, maybe, nobody, really, stuff
123. anything, funny, guy, it's, maybe, pretty, really, stuff, stupid, thing
124. anything, guy, it's, maybe, stupid, thing, wrong
125. anything, it's, nothing, wrong
126. anywhere, cyclone
127. arguably, brilliant, exciting, memorable
128. arguably, certainly, exciting
129. arguably, ever, exciting
130. arguably, good
131. arguably, great
132. ass, bad, guy, stupid, ugly
133. atrocity, bad, famine, terrible
134. atrocity, bad, horrible, terrible
135. attendance, crowd, funeral
136. awful, bad, horrible, it's, joke, really, stuff, stupid, terrible
137. awful, bad, horrible, smell, terrible
138. awful, bad, horrible, stuff, stupid, terrible, ugly
139. awful, bad, it's, joke, pretty, really, stuff, stupid, terrible

140. awful, bad, notoriously
141. awful, bad, pretty, stuff, stupid, terrible, ugly
142. awful, horrible, it's, really, terrible, truly
143. bad, behaviour, habit
144. bad, behaviour, offender
145. bad, breath, smell
146. bad, condition, drought, terrible, weather
147. bad, condition, visibility, weather
148. bad, cop, guy, stupid
149. bad, deed, evil, good
150. bad, deed, evil, horrible, terrible
151. bad, deed, evil, karma
152. bad, disaster, drought, famine, terrible
153. bad, drought, famine, harvest
154. bad, drought, flood, flooding, weather
155. bad, drought, harvest, weather
156. bad, drought, recession
157. bad, ending, joke
158. bad, ending, mood
159. bad, ever
160. bad, evil, good, luck
161. bad, evil, horrible, nightmare, terrible
162. bad, evil, horrible, terrible, ugly
163. bad, evil, luck, omen
164. bad, evil, luck, terrible
165. bad, good, guy, pretty
166. bad, good, showing
167. bad, guy, it's, joke, pretty, really, stuff, stupid
168. bad, guy, it's, pretty, really, stuff, stupid, thing

169. bad, guy, pretty, stuff, stupid, thing, ugly
170. bad, habit, taste
171. bad, habit, temper
172. bad, harvest, luck, timing
173. bad, horrible, it's, really, stuff, stupid, terrible, thing
174. bad, horrible, mistake, stupid, terrible
175. bad, horrible, nightmare, stuff, terrible
176. bad, horrible, stuff, stupid, terrible, thing, ugly
177. bad, horrible, temper, terrible, ugly
178. bad, it's, mood
179. bad, it's, pretty, really, stuff, stupid, terrible, thing
180. bad, mood, temper, ugly
181. bad, notoriously, temper
182. bad, pretty, stuff, stupid, terrible, thing, ugly
183. bad, progressively, worse
184. bad, reputation, temper
185. bad, situation, worse
186. bad, smell, taste
187. bad, thing, worse
188. badly, crash, recession
189. badly, flood, flooding
190. badly, wrong
191. beautiful, brilliant, it's, perfect, pretty
192. beautiful, it's, pretty, terrible, thing
193. beautiful, it's, terrible, truly
194. beautiful, jealous, ugly
195. beautiful, pretty, terrible, thing, ugly
196. behaviour, drunk, offender
197. behaviour, neighbour

198. beloved, companion, loyal
199. beloved, companion, narrator
200. beloved, comrade, dear, farewell, friend, goodbye
201. beloved, comrade, deceased
202. beloved, comrade, friend, loyal
203. beloved, deceased, lover
204. beloved, farewell, forever, goodbye
205. beloved, farewell, friend, lover
206. beloved, friend, girlfriend, jealous, lover
207. beloved, friend, mentor
208. beloved, girlfriend, jealous, lover, secretly
209. beloved, loyal, secretly
210. biographer, certainly, doubt, hardly
211. biographer, generous
212. birthday, dinner, wedding
213. boyfriend, classmate, co-worker, crush, friend, girlfriend, jealous, roommate
214. boyfriend, classmate, co-worker, crush, friend, jealous, neighbor, roommate
215. boyfriend, classmate, crush, friend, friendship, jealous, roommate
216. boyfriend, classmate, crush, friendship, teenage
217. boyfriend, classmate, crush, girlfriend, teenage
218. boyfriend, classmate, crush, neighbor, teenage
219. boyfriend, cop, girlfriend, guy
220. boyfriend, crush, girlfriend, guy, jealous
221. boyfriend, crush, girlfriend, jealous, secretly
222. boyfriend, dinner, drunk, girlfriend, jealous
223. boyfriend, dinner, drunk, wedding
224. boyfriend, dinner, friend, girlfriend, jealous
225. boyfriend, drunk, stranger
226. boyfriend, friend, girlfriend, jealous, lover

227. boyfriend, girlfriend, jealous, lover, secretly
228. boyfriend, girlfriend, vacation
229. breath, drunk
230. brilliant, concerto
231. brilliant, feat, memorable, try
232. brilliant, feat, perfect
233. brilliant, reputation
234. certainly, exciting, it's, maybe, pretty, stuff
235. certainly, maybe, perhaps
236. certainly, unlikely
237. childhood, classmate, crush, friendship, teenage
238. childhood, classmate, crush, neighbor, teenage
239. childhood, friend, imaginary
240. childhood, narrator, nightmare
241. childhood, narrator, teenage
242. circle, imaginary
243. close, collaborator, friend
244. close, confidant, friend
245. co-star, friend, girlfriend, jealous
246. co-star, friend, girlfriend, longtime
247. co-star, funny
248. co-worker, joke
249. collaborator, friend, lifelong, mentor
250. collaborator, friend, long-time, mentor
251. collaborator, friend, longtime, mentor
252. colleague, friend, lifelong, mentor
253. colleague, friend, long-time, mentor
254. colleague, friend, longtime, mentor
255. companion, dwarf

256. companion, intimate, lifelong
257. companion, long-time
258. companion, longtime
259. comrade, deceased, funeral
260. comrade, farewell, funeral
261. comrade, nightmare
262. concerto, extant, known
263. condition, cure, terrible
264. condition, drought, temperature, weather
265. confidant, dear, friend
266. confidant, friend, intimate, lifelong
267. confidant, friend, lifelong, mentor
268. confidant, friend, long-time, mentor
269. confidant, friend, longtime, mentor
270. cop, guy, tough
271. crowd, crush
272. crowd, defeat, try, victory
273. crowd, farewell, funeral
274. crowd, farewell, gig
275. crowd, heel
276. cultivar, drought, harvest
277. cultivar, dwarf, known
278. cultivar, habit, taste
279. cultivar, known, specimen
280. cure, deadly, terrible
281. cure, known, treatise
282. cyclone, deadly, disaster, fatality, flood, flooding, tornado
283. cyclone, deadly, fatality, flood, recorded, tornado
284. cyclone, drought, flood, flooding, weather

285. cyclone, drought, flooding, occurrence
286. cyclone, drought, temperature, weather
287. cyclone, fatality, flooding, occurrence, tornado
288. cyclone, flood, flooding, tornado, weather
289. cyclone, known, occurrence
290. cyclone, known, recorded
291. cyclone, recorded, temperature
292. deadly, disaster, eruption, famine, fatality, flood
293. deadly, disaster, famine, fatality, flood, flooding
294. deadly, disaster, famine, terrible
295. deadly, eruption, fatality, flood, recorded
296. deadly, evil, foe, terrible
297. deadly, parasite
298. dear, friend, neighbour, stranger
299. deceased, inscription
300. deceased, relative
301. deed, evil, forever, terrible
302. deed, generous
303. deed, mention, written
304. deed, poll
305. defeat, hat-trick, try, victory, win
306. definite, earliest, known, mention
307. definite, ending
308. definite, hardly
309. definite, imaginary
310. definite, known, occurrence
311. definite, nothing
312. dinner, farewell, friend
313. dinner, friend, intimate

314. dinner, recipe
315. disaster, drought, eruption, famine, flood
316. disaster, famine, fatality, tragedy
317. disaster, famine, terrible, tragedy
318. disaster, recipe
319. drought, eruption, occurrence
320. drought, eruption, temperature
321. drought, postseason
322. drought, premiership, trophy
323. drunk, funny, stupid
324. drunk, offender, stranger
325. dwarf, known, lesser
326. dwarf, known, tapeworm
327. dwarf, temperature
328. dwarf, ugly
329. earliest, extant, inscription, known, mention, written
330. earliest, extant, known, printed, treatise, written
331. earliest, known, mention, recorded
332. earliest, known, printed, recipe
333. ending, funny, joke
334. ending, goodbye, happily
335. ending, mood, perfect
336. ending, tragedy
337. eruption, fatality, occurrence
338. eruption, known, recorded
339. eruption, recorded, temperature
340. ever, exciting, funny
341. ever, feat, hat-trick
342. ever, feat, postseason

343. ever, happily
344. evil, forever, nightmare, terrible
345. evil, horrible, terrible, truly
346. evil, jealous, ugly
347. evil, lesser
348. evil, wrong
349. exciting, finish
350. exciting, funny, it's, maybe, pretty, really, stuff, thing
351. exciting, memorable, overtime
352. farewell, goodbye, luck
353. feat, hat-trick, memorable, try, win
354. feat, hat-trick, premiership, try, win
355. feat, hat-trick, remarkable
356. feat, perfect, postseason, undefeated
357. feat, postseason, undefeated, win
358. feat, premiership, trophy, win
359. feat, premiership, undefeated, win
360. finish, good, showing
361. finish, victory, win
362. foe, friend, imaginary
363. foe, friend, long-time
364. foe, friend, longtime
365. foe, friend, stranger
366. forever, friendship
367. forever, goodbye, happily
368. forever, stranger
369. fossil, record
370. friend, good
371. friend, loyal, stranger

372. friendship, unlikely
373. funeral, wedding
374. funny, guy, it's, joke, nobody, really, stuff
375. funny, guy, it's, joke, pretty, really, stuff, stupid
376. funny, guy, it's, pretty, really, stuff, tough
377. funny, horrible, it's, joke, really, stuff, stupid, terrible
378. funny, horrible, it's, really, stuff, stupid, terrible, thing
379. funny, horrible, it's, really, terrible, truly
380. funny, it's, joke, pretty, really, stuff, stupid, terrible
381. funny, it's, pretty, really, stuff, stupid, terrible, thing
382. funny, timing
383. generous, temper
384. gig, intimate
385. gig, memorable
386. girlfriend, narrator, teenage
387. good, known
388. good, selling
389. goodbye, happily, luck
390. great, lesser
391. great, recession
392. guy, jealous, ugly
393. guy, rudo, rudos
394. habit, lifelong
395. half-life, isotope, known, meteorite
396. happily, lover
397. happily, luck, wedding
398. hat-trick, memorable, overtime, victory, win
399. hat-trick, memorable, try, victory, win
400. hat-trick, premiership, try, victory, win

401. horrible, terrible, tragedy
402. intimate, it's, mood
403. isotope, relative
404. isotope, temperature
405. it's, mood, perfect
406. jealous, temper, ugly
407. known, parasite, tapeworm, vertebrate
408. known, perhaps
409. lesser, offender
410. memorable, perhaps
411. mistake, stupid, wrong
412. occurrence, unlikely
413. offender, tough
414. perfect, stranger
415. perfect, timing
416. poll, recession
417. poll, showing
418. premiership, trophy, victory, win
419. premiership, undefeated, victory, win
420. recipe, taste
421. record, written
422. reputation, tough
423. rudo, rudos
424. rudo, técnicos
425. teenage, tragedy
426. tough, undefeated
427. undefeated, wrestler

References

[1] MILLER G A. WordNet: a lexical database for English[J]. Communications of the ACM, 1995, 38(11): 39–41.

[2] LIU H, SINGH P. Focusing on conceptnet's natural language knowledge representation[C]//Proc. of the 8th Intl Conf. on Knowledge-Based Intelligent Information and Engineering Syst. [S.l.]: [s.n.], 2004.

[3] LANDAUER T K, DUMAIS S T. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.[J]. Psychological review, 1997, 104(2): 211.

[4] DEERWESTER S, DUMAIS S T, FURNAS G W, et al. Indexing by latent semantic analysis[J]. Journal of the American society for information science, 1990, 41(6): 391.

[5] HYUNGSUK J, PLOUX S, WEHRLI E. Lexical knowledge representation with contexonyms[C]//9th MT summit Machine Translation. [S.l.]: [s.n.], 2003: 194–201.

[6] MIKOLOV T, YIH W.-T, ZWEIG G. Linguistic Regularities in Continuous Space Word Representations.[C]//Hlt-naacl. Vol. 13. [S.l.]: [s.n.], 2013: 746–751.

[7] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. ArXiv preprint arXiv:1301.3781, 2013.

[8] PENNINGTON J, SOCHER R, MANNING C D. Glove: Global Vectors for Word Representation.[C]//EMNLP. Vol. 14. [S.l.]: [s.n.], 2014: 1532–1543.

[9] FADAEE M, BISAZZA A, MONZ C. Learning Topic-Sensitive Word Representations[J]. ArXiv preprint arXiv:1705.00441, 2017.

[10] LESK M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone[C]//Proceedings of the 5th annual international conference on Systems documentation. ACM. [S.l.]: [s.n.], 1986: 24–26.

[11] NAVIGLI R, VELARDI P. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation[J]. IEEE transactions on pattern analysis and machine intelligence, 2005, 27(7): 1075–1086.

[12] IACOBACCI I, PILEHVAR M T, NAVIGLI R. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity.[C]//ACL (1). [S.l.]: [s.n.], 2015: 95–105.

[13] MORO A, RAGANATO A, NAVIGLI R. Entity linking meets word sense disambiguation: a unified approach[J]. Transactions of the Association for Computational Linguistics, 2014, 2: 231–244.

[14] YAROWSKY D. Unsupervised word sense disambiguation rivaling supervised methods[C]//Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics. [S.l.]: [s.n.], 1995: 189–196.

[15] BENZÉCRI J.-P. Correspondence analysis handbook[M]. [S.l.]: Marcel Dekker, 1992.

[16] TOMITA E, TANAKA A, TAKAHASHI H. The worst-case time complexity for generating all maximal cliques[C]//International Computing and Combinatorics Conference. Springer. [S.l.]: [s.n.], 2004: 161–170.

[17] WANG R, ZHAO H, PLOUX S, et al. A bilingual graph-based semantic model for statistical machine translation[C]//International Joint Conference on Artificial Intelligence. [S.l.]: [s.n.], 2016.

[18] PLOUX S, JI H. A model for matching semantic maps between languages (French/English, English/French)[J]. Computational linguistics, 2003, 29(2): 155–178.

[19] EDMONDS P, HIRST G. Near-synonymy and lexical choice[J]. Computational linguistics, 2002, 28(2): 105–144.

[20] SCHMID H. Improvements in part-of-speech tagging with an application to German[C]//In proceedings of the acl sigdat-workshop. Citeseer. [S.l.]: [s.n.], 1995.

[21] SCHMID H. Probabilistic part-ofispeech tagging using decision trees[C]//New methods in language processing. Routledge. [S.l.]: [s.n.], 2013: 154.

[22] SANTORINI B. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision)[J]. 1990.

[23] BRON C, KERBOSCH J. Finding all cliques of an undirected graph (algorithm 457)[J]. Commun. ACM, 1973, 16(9): 575–576.

[24] KOCH I. Enumerating all connected maximal common subgraphs in two graphs[J]. Theoretical Computer Science, 2001, 250(1-2): 1–30.

[25] LEE L, PEREIRA F. Distributional similarity models: Clustering vs. nearest neighbors[C]//Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. Association for Computational Linguistics. [S.l.]: [s.n.], 1999: 33–40.

[26] HINTON G E. Distributed representations[J]. 1984.

[27] BENGIO Y, DUCHARME R, VINCENT P, et al. A neural probabilistic language model[J]. Journal of machine learning research, 2003, 3(Feb): 1137–1155.

[28] MORIN F, BENGIO Y. Hierarchical Probabilistic Neural Network Language Model.[C]//Aistats. Vol. 5. Citeseer. [S.l.]: [s.n.], 2005: 246–252.

[29] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. [S.l.]: [s.n.], 2013: 3111–3119.

[30] NAVIGLI R, PONZETTO S P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network[J]. Artificial Intelligence, 2012, 193: 217–250.

[31] YAO X, VAN DURME B. Nonparametric bayesian word sense induction[C]//Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing. Association for Computational Linguistics. [S.l.]: [s.n.], 2011: 10–14.

[32] LE Q, MIKOLOV T. Distributed representations of sentences and documents[C]//Proceedings of the 31st International Conference on Machine Learning (ICML-14). [S.l.]: [s.n.], 2014: 1188–1196.

[33] FENK-OCZLON G, FENK A. The association between word frequency and polysemy: a chicken and egg problem[C]//Proceedings of the XIIth International Conference "Cognitive Modeling in Linguistics", Kazan State University Press, Kazan. [S.l.]: [s.n.], 2010: 167–170.

[34] MCCARTHY D, NAVIGLI R. Semeval-2007 task 10: English lexical substitution task[C]//Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics. [S.l.]: [s.n.], 2007: 48–53.

[35] ILLES J, FRANCIS W S, DESMOND J E, et al. Convergent cortical representation of semantic processing in bilinguals[J]. Brain and language, 1999, 70(3): 347–363.

[36] MIKOLOV T, LE Q V, SUTSKEVER I. Exploiting similarities among languages for machine translation[J]. ArXiv preprint arXiv:1309.4168, 2013.

Acknowledgement

This thesis was developed in a short duration of three months. But the preparation took four years of serious academic effort.

First I would address my thanks to my supervisors Prof. Sabine Ploux of CNRS and Prof. Bao-Liang Lü of SJTU for the thesis proposal. This thesis provides me with a guidance through the domain of language modeling, machine learning and information processing. Their timely responses to my questions and mails were helpful during the phase of conception of models and actual coding. They examine in detail my reports and the experiment results, and offer their thoughtful suggestions on the content of the report and the possible improvements to the experiment set up.

Prof. Ploux introduced me to the interesting application of graph theory into computational linguistic study. The demonstrations made by Prof. Ploux, and her recommended courses and articles provided indispensable knowledge for me to commence this study. She provided key elements of the filter configuration during contexonym association table construction, and offered her help to validate the quality of contexonyms and cliques.

Many thanks to my co-supervisor Prof. Hai Zhao of SJTU. Prof. Zhao is very involved in the thesis research. Together with Prof. Ploux, we had rich exchanges on the methodology and the orientation of the research and the experiment. Prof. Zhao also provided many intelligent insights on data source selection, machine learning system configuration and appropriate evaluation. Prof. Zhao also provided the computational resource for this thesis.

Many thanks to my long-term advisor Prof. Jialiang Lu of SPEIT. It is Prof. Lu who introduced me to Prof. Ploux. He offered countless help and insights on academic suggestions, research methodologies and hardware supports since the first year I entered SJTU, to which I am extremely grateful.

I wish every one who provided me with supports and suggestions the best. An exhaustive list could not be made, but special thanks to Prof. Alain Chillès, Yue Bi, Yuxiang Gong, Muyang

Yi, Mengxue Zhang, Xuan Liao, Zeyuan Wu, Yifan Jia, Xiao Feng, Binghan Jin.

图论与深度学习结合的语义表示

在自然语言处理相关研究中，分布式向量表示法是将单词信息编码的常用方法。这种表示方法克服了经典方法中以单词在词汇表中的位置进行编码带来的高维度问题，以连续空间中的一个实数向量为依托，并进一步提出了以向量度量为基准的潜在信息的计算方法。大量神经语言模型使用分布式向量表示法总结计算其系统内处理之文本的抽象信息。诸如 Word2Vec 和 GloVe 之类的分布式向量表示训练方法近年在学界大受欢迎。以此方法建立的词嵌入空间能保留语言学意义上的单词间规律和语义相对关系，因此这些词嵌入模型在自然语言测试中取得了惊人的成果。

这些以单词为基本单元的嵌入系统的缺点之一，在于语义和词语并非存在一一对应的关系。人类对于语言词汇的选择基于词义，而不是词形。为了建立更准确的语言模型，相比于单词，词义是嵌入连续空间实体的更优候选者。

词义嵌入的现有文献多采用受监督或半监督学习的方法。他们或依靠大量结构明确的手工数据库来构建一个语义网，要么需要人工对语义项进行聚类和规则编码。本论文提出了无监督环境下，使用子元素相邻的完全子图（上下文词团 contexonym clique）来从语义层面对文本进行消歧义的方法。我们利用从大型单语语料库中获取的单词词频和单词对在同句出现的频率等统计信息，构建一个以单词为顶点，以词和词间的相互关联性为边的图结构。为了获取随机单词的上下文词团集合，我们通过对其构建上下文词组 (contexonym) 集合和以词间关系图结构为基准的图边抽取，从新生成的小型图结构中提取最大完全子图的集合作为上下文词团集合。

图论的完全子团抽取问题是 21 个不能使用多项式复杂度算法解决的问题。为了保证上下文词团的生成速度，本文中采用了动态选取上下文词组集合和图密度自动调整的方法。此种方法同时也保证了词团的数量、词团中上下文词组的个数落在合理区间。

在本论文中，上下文词团被视作词义的最小表述单元。这些上下文词团被用于对文本中的对应单词打上词义标记从而起到系统内消歧义的作用。词团的选择基于被消歧义单词实例所处的上下文语境：在上下文中出现的词条总集是被消歧义单词所具备的上下

文词团集合选择词团的判断基准。我们使用了预先训练完成的词嵌入向量衡量此上下文实例中的词组与当前受消歧词汇所具有的上下文词组间的相似度，通过生成一条临时的伪上下文词团，以词团间的 χ^2 距离为词团选择依据。此词义标记代表的语义含义是通过该单词上下文词团中的上下文词组体现的，该词团在连续语义嵌入空间中的向量与其他向量的相对空间关系也是词义含义的来源之一。

此论文使用基于 Skip-Gram 修改的训练方案，以被上下文词团标记从而被消歧义过的语料库为数据来源来学习语义嵌入表示。在训练过程中，我们利用以中间词汇和其被标记的词团的双栈向量表示为基础，通过映射到神经网络隐层，以通过层次化 Softmax（Hierarchical Softmax）形成其对上下文窗口中词汇预测的概率分布。此概率分布为训练目标表述，中间词汇在理想条件下应产出除上下文词汇外其他词汇概率均为 0 的分布。此方法同时训练词汇向量和词团向量，词团向量则为嵌入的语义向量。我们同时以段落嵌入的方法训练了以上下文词团成员词组为数据来源的词团向量，并与其他系统以词团相似度为标准进行了比较。

我们使用英语词汇替代任务（Lexical Substitution）对基于上下文词团消歧义系统和语义嵌入系统进行了测试。该测试令受试模型自行判断目标单词在句中的含义，并从模型自身的词汇库中选取相近的词汇或词组替代目标单词。根据评估结果显示，我们的模型系统能厘清不同上下文间的单词词义，并且对环境进行了细微的区分。在嵌入系统中获取的词义嵌入向量能还原有效的词汇，并且词义间的相关性也在嵌入系统中得到了保留。我们对该模型与其他依据传统模型在相同条件下训练得到的嵌入空间进行了比较。比较结果揭示我们的词义嵌入方案比其他方案更为有效。