Towards aspectual classification of clauses in a large single-domain corpus

Qiwei Peng

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2018

Abstract

Researchers are beginning to develop large, single-domain corpora, either text alone (such as instructions, recipes, tweets, etc.) or multi-media (image captions, video captions, etc.). The language used in such corpora tends to share properties across different texts. The aim of this project was to explore whether a compositional distributional semantics model could be effective in classifying the aspectual properties of clauses in a large corpus of instructions. However, Gold Standard annotation of aspectual properties did not exist for the recipe instruction corpus we planned to use, the project ended up comprising two preliminary tasks: 1) determining the extent to which a compositional distributional semantics model could be used in classifying aspectual properties of clauses in small mixed text corpora (SitEnt and telicity dataset), for which Gold Standard annotation did exist. 2) developing a systematic way of reducing the effort required to produce Gold Standard annotation of a large single-domain corpus (MIT recipe corpus), by carrying lexical and semantic deduplication, so that only one token from each equivalence class in the resulting deduplicated corpus needs to be annotated.

we have conducted experiments on the aspectual classification problem with two different distinctions, state/event (e.g. to know/swim or walk) and telic/atelic (whether it has a natural ending point), by exploiting self-trained word embeddings and existing compositional models on the SitEnt/telicity corpus. Our results have shown that by only considering distributional features, the compositional model achieves the state-of-the-art results. By exploring several comparisons, we find that the *auxiliary* is important when making the decision between event and state while *noun subject* and *direct object* seem to contribute more if considering the distinction between telic and atelic. From our experiments, the simple addition model is shown to be more effective at capturing semantic interactions compared to the practical lexical function model. Due to the need of deduplication for the MIT recipe dataset on the aspectual classification task, we have also proposed a stable deduplication pipeline by utilizing the LSH and LSA model.

Acknowledgements

I would like to thank my supervisor, Prof. Bonnie Webber and Dr. Thomas Kober, for their consistent support and patience throughout the course of the project. A big thank you to my flatmate and friend, Bowen Wang, who helped me with his endless patience and advice.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Qiwei Peng*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Properly representing the meaning of phrases or sentences is a challenging task and is always the key point for understanding natural languages. The semantics expressed by a sentence involves complex interactions among different sentence components. Methods like constructing logical forms are exploited to attempt to accurately represent interactions like negation scope [Fancellu et al., 2017]. Logical forms are widely used in NLP tasks like question answering system [Moldovan and Rus, 2001], relation entailment [Lascarides and Asher, 1993] and statistical machine translation [Quirk, 2004]. However, there are aspects of semantics that are ignored in representations of logical forms [FitzGerald et al., 2013]. Rather than attempting to incorporate them in the logical form and taking advantage of it, many work just construct linguistic features (like tense, part of speech, aspect etc.) to include this semantic information [Collins et al., 2017].

Distributional word vectors or word embeddings, which are obtained from corpus statistics, have been shown to be useful in representing word meanings for last two decades. After the birth of word2vec [Mikolov et al., 2013], word vectors begins to play a dominate part in various neural network based architectures. The most famous example of word vectors capturing semantics is: **King - Man + Woman = Queen**. Due to the empirically successful ability of capturing both syntactic and semantic information, word vectors now are widely used in various NLP tasks as the fundamental word representation. After getting a good representation for word meanings, it is quite natural to think about obtaining phrase and sentence meanings in the same manner by composing word vectors. Neural networks have been exploited to get the sen-

tence meaning representation by compressing the concatenated word representations to lower dimensional vector representation [Zheng et al., 2018]. Though this method is widely used, it needs considerable labelled data to get the model properly trained and the trained model is task-specific which means it can not be transferred easily. A more general and common method of obtaining sentence representation via word representation is the compositional distributional semantics model (CDSM). Various CDSMs have been proposed to take advantage of word vectors and bring in syntactic and semantic information [Mitchell and Lapata, 2008, Blacoe and Lapata, 2012, Paperno et al., 2014, Weber et al., 2017]. Such models follow specific combination rules and could be easily modified to accommodate specific needs. Therefore, CDSMs are desirable tools to investigate specific language phenomena. To further explore capabilities of CDSMs, this project has applied them to aspectual classification, to assess if it can capture semantic relationships that underpin aspectual classification decisions.

Clauses in text have different aspectual properties and thus play different roles in the discourse. Aspectual classification maps clauses into a small set of categories for reasoning about time. For instance, *I remember your father* is an event, distinguished from a state, like *you resemble your father*. Further detailed classification (activities, accomplishments and achievements) can be made within the event class according to two dimensions: one is whether it indicates a natural ending point, and the other is the contrast between punctuality and temporal extension [Dowty, 1979]. The distinction is mainly affected by the property of main verb and the context it occurs in. In other words, it can be reflected by the semantics of the clause. This classification is necessary for interpreting temporal relationships [Moens and Steedman, 1988] and is a required component for many NLP applications that need the ability to reason about time like temporal processing and information extraction [Siegel and McKeown, 2000, Costa and Branco, 2012].

To explore such a classification problem, one must understand the meaning of the clause. When it comes to the NLP area, it means that the meaning (semantics) of the clause should be properly represented before feeding to a classifier. So it could also be recognised as a clause meaning representation problem. The initial motivation of the project is to investigate the ability of CDSMs and the effectiveness of the distributional vectors on representing both syntactic and semantic information. To explore that, we assess the quality of composed meaning representation on the aspectual type classification problem where the decision is directly determined by the semantics of the clause. Most previous investigations on this classification problem did not exploit

word vectors, they mainly brought in semantic information by including linguistic features and context words [Siegel and McKeown, 2000, Friedrich and Palmer, 2014]. In this project, word vectors will be used as the only feature for classification to explore the effect of compositional models.

One task we want to deal with is to explore potential differences of aspectual type that might have on different sentence structures (e.g. imperative instructions and declarative news). The SitEnt dataset [Friedrich et al., 2016] and telicity dataset [Friedrich and Gateva, 2017], which mainly consist of declarative sentences, are already annotated in terms of aspectual class, while the MIT recipe instruction dataset [Salvador et al., 2017], which is planned to be used for control experiments, is unlabelled. To train a task-specific classifier, a lot of annotation work on the instruction dataset need to be done.

The MIT recipe dataset is made of cooking instructions, and therefore has a large number of instructions that are syntactically or semantically similar. For instance, the instruction *preheat oven to 150 degree* does not have different aspectual properties than the instruction *preheat the oven to 300 degree* in terms of aspectual property, and the instruction *Shape a dough into a plait* has the same aspectual type with *Form the pastry into a ball*. There is little benefit in having these instructions annotated separately and include them in the training data. Because firstly, these repeated sentences are not helpful in training the aspectual type classifier, it is like keeping oversampling the same class, secondly labelling all these sentences would waste the annotator's time by doing the same work for they actually share the same annotation. Thus, it is necessary to identify what sentences do not have to be annotated, since they are similar to other sentences, only one of them needs to be annotated. Due to the large number of these similar sentences, performing near-duplicates detection can significantly reduce the size of corpus and relief the required work for annotation. Due to the time limit, the project only investigates the deduplication pipeline on MIT recipe corpus and leaves the following annotation and classification problem to future work.

1.2 Objectives

There are two main objectives in this project:

- Aspectual classification: To assess the ability of CDSMs and the effectiveness of distributional vectors on representing syntactic and semantic information. Two

different aspectual distinctions (event/state and telic/atelic) will be investigated by using simple addition and the practical lexical function model.

- Corpus deduplication: Since necessary annotation work need to be done on the MIT recipe dataset, deduplication process is desired to significantly reduce the corpus size and relief the annotation work. Therefore, We plan to find out a stable deduplication process that could maximumly detect near-duplicates that are semantically similar and have high lexical overlap.

1.3 Contributions

This project mainly makes two contributions: 1) our experiment results show that CDSMs could effectively model the aspectual class of clauses, and we find that the *auxiliary* contributes the most when making decision between state and event while the *noun subject* and *direct object* seem to be more informative when considering the distinction between telic and atelic. 2) by utilizing the LSH and LSA model, we propose a stable deduplication process for detecting lexical and semantic near-duplicates on a large single-domain corpus (MIT recipe corpus).

1.4 Thesis structure

Chapter 1 is the introduction. It describes the motivation of the project, research problems the project hopes to investigate and the general structure.

Chapter 2 gives a detailed description of the background of the project, it introduces aspectual types, compositional distributional semantics models, document clustering techniques like locality sensitive hashing (LSH) and latent semantic analysis (LSA) which have been used in the deduplication process. It also surveys the relevant literature on aspectual classification problem and compositional modelling, exploring some existing approaches.

Chapter 3 describes two main experiments that are used to investigate the aspectual classification with different distinction (state/event and telic/atelic). Introduction of datasets, preprocessing steps and experiment settings are presented. Experiment results and relevant analysis are also reported in this chapter.

Chapter 4 describes the whole deduplication process step by step and evaluates its stability on various size of the MIT recipe corpus.

Chapter 5 concludes the project. It summarises the experiment results and analysis. Pointing out the experiment limitations and works that are not achieved due to time and computing resource limits. It also gives directions for future work.

Chapter 2

Background

In this chapter, we will describe some important background knowledge. In section 2.1, the aspectual class problem will be explained in detail. Then section 2.2 summarises well-known compostional models to see how they could be used in representing clause meanings. In section 2.3, we give introduction of the corpus deduplication task and its relation to aspectual classification problem.

2.1 Aspectual type

2.1.1 Aspectual type distinction

Verbs can be classified into different categories according to their logical entailments, interactions with temporal modifiers and tense. Dowty [1979] and Vendler [1967] proposed a commonly accepted classification of four types, namely achievements, activities, accomplishments and states. In this classification, two properties are considered as crucial in making the decision, the first one is whether it has a natural ending point (whether it is telic) and the second one is whether it involves the change of states (whether it has stages or whether it could occur with the progressive tense). Achievements, activities and accomplishments are also classified together as event class which is used to distinguish from the state. Moens and Steedman [1988] also specifies a *point* class which was used to be incorporated in the activity class in Dowty's classification. This separation is advocated by Smith [1991] who claims that point class is atelic achievements and should not be included into the activity class. More classification criteria have been proposed and discussed [Smith, 2003, 2005, Palmer et al., 2007] regarding different models of discourse, while in this project, we mainly con-

sider the classification proposed by Dowty. Dowty [1979] provides a list of example verbs and a detailed criterion for each class, as shown in Table 2.1. Table 2.1 shows that states are atelic (-telic) and non-dynamic (have no stages). It has no stages for there is no change takes place when the state holds. Activities are atelic, which is the same as states, and dynamic, which is different from states. Verbs in this category can be combined with *for*-durational phrases but not *in*-phrases. For example, it is natural to say *I walked for an hour* rather than *I walked in an hour*. Accomplishments are telic and dynamic. Intuitively, accomplishments are activities that move towards an ending point. Being different from activities, it is natural for accomplishments to be combined with *in*-phrases rather than *for*-phrases like *I painted a picture in 2 weeks* rather than *I painted a picture for 2 weeks*. Like accomplishments, achievements have natural ending point while they are non-dynamic like states, which means that it is not natural to combine them with progressive tense.

	+stage	-stage
+telic	**Accomplishments** build a house, paint a picture	**Achievements** reach, notice, recognise
-telic	**Activities** walk, swim, run	**States** believe,have, resemble

Table 2.1: Dowty's aspectual classification and example verbs

Examples of verbs in Table 2.1 indicate the typical types they bear. Like *believe* is often assumed to represent a state and the sentence *I believe you* is also considered as a state for it does not have a natural ending point and it also does not involve the change of states. However, we could not assume that the aspectual class of a sentence could always be revealed by that of its verb in isolation. Aspectual class of the verb, which is sometimes called lexical aspect, might not be compatible with that of the same verb in context (or a sentence) [Moens and Steedman, 1988]. For instance, though *build* often represents accomplishments and sentence *I built a house* is also an accomplishment, but the sentence *I was building a house* becomes an activity. From this example, we could see that the aspectual class of a clause is not solely determined by the verb but also its context. We could also say that the aspectual class of the clause is the property of a verb in its context. Moens and Steedman deeply investigate the phenomenon of

aspectual type transiting from one class to another, which is caused by various linguistic devices like aspects, tenses, temporal adverbials and auxiliaries posing constraints on the main verb. This kind of transformation is called aspectual coercion or aspectual transformation.

2.1.2 Aspectual classification problem

As discussed in above section, the aspectual type of a clause is reflected by the main verb in its context since various linguistic devices could pose constraints or transformations on the aspectual class of the verb and make transition from one class to another. It is quite natural to use linguistic features observed in the clause to represent the transformation. This feature or indicator selection is often done manually and then used to tackle the problem. Siegel and McKeown [2000] point out that, though individual linguistic indicators have predictive value, they are not deterministic. To decide the aspectual class of a clause, one should take full advantage of complete range of linguistic indicators by coordinating and combining them. Researchers then turn to machine learning techniques for they can be employed to automatically produce a model that could combine all linguistic indicators.

Siegel and McKeown [2000]'s approach is the early work that applies the machine learning methods (like logistic regression and decision tree) to the aspectual type classification. They mainly discuss the classification problem at clause level and make distinction between state and event as well as telic and atelic events. The aspectual class they focus on is the fundamental aspectual class, which refers to the basic class before any aspectual transformation or coercion is applied. Therefore, the fundamental aspectual class is a function of the main verb and a select group of complements. To properly explore the fundamental aspectual class, Siegel and McKeown exploit 14 numeric corpus-based linguistic indicators. Several indicator examples are given in Table 2.2. Each indicator has a numeric value indicating that how frequently the main verb occurs in a clause with a linguistic marker as listed in Table 2.2. For example, the indicator *temporal adverb* for a verb measures the proportion of the verb's occurrence in the corpus together with a *temporal adverb* such as *frequently* and *then*. This measurement makes sense for we have discussed in section 2.1.1 that each aspectual type has its own unique features such as accomplishments are more often to be combined with *in*-phrase while activities are more often to be combined with *for*-phrase. When counting how frequently it occurs with these patterns, we could get a

general idea of its fundamental aspectual type. By exploring these linguistic features with machine learning methods, [Siegel and McKeown, 2000] achieve high accuracy on clauses from medical discharge summaries. But it actually fails to outperform the baseline of memorizing the most frequent class of the verb type.

Linguistic Indicator	Example Clause
temporal adverb	I saw to it **then**
duration *in*-PP	She built it **in an hour**
progressive	They **have** landed
no subject	He was admitted to the Hospital
...	...

Table 2.2: Four Siegel and McKeown's linguistic indicator examples

Later, Friedrich and Palmer [2014] proposed a new model to tackle the problem. As opposed to using the fundamental class, they predict the aspectual type of the clause by predicting the type of the verb in context of arguments and modifiers. In other words, the aspectual class they care about is the one after transformations are applied. Representations used in this model are also feature-based. In addition to linguistic features used by Siegel and McKeown [2000], they further extend to include distributional features (word vector for each verb type) and instance-based features. Instance-features used here are non-numeric and not extracted from background corpus. They take categorical values solely based on current clause (e.g. tense: past, perfect: true, voice: passive). By combining more informative features, their work shows that the aspectual class prediction with context performs better on their own-constructed corpus particularly when dealing with verbs with ambiguous aspectual types. Their work also reveal the syntactic and semantic capturing ability of distributional word embeddings. When solely exploiting the distributional features, the model is only slightly worse than the best performed model which takes advantage of all three different kinds of feature types. Friedrich and others later conduct various experiments on tackling similar aspectual classification problem in different distinctions (e.g. static, habitual and episodic) considering the contribution to the discourse [Friedrich and Pinkal, 2015] and constructing more complex models by exploring potential informative linguistic cues [Friedrich et al., 2016]. Costa and Branco [2012] estimates aspectual indicators by collecting specific google search hits because their lack of syntactic information. Though the performance is poor, it proves that aspectual class information is quite

helpful for tasks like temporal relation processing.

Though various models and informative features have been explored to automatically identify the aspectual class of a clause, they are mainly feature-based as mentioned above. Those features are extracted manually and therefore are very time-consuming. As discussed above, Friedrich and Palmer [2014]'s experiment reveals that distributional word embeddings for verbs have good ability of capturing necessary information for identifying aspectual class. Therefore, this gives us inspiration that we might not only employ distributional features for verbs, but also the context.

2.2 Compositional distributional semantics models

As stated in above section, the key point of representing the aspectual class of a clause is to reflect the property of the main verb in context. In other words, the context which the verb occurs is crucial and should be represented properly. Apart from manually constructing linguistic or instance-based features, another way to leverage the context information is to compose the verb with its context together by exploiting distributional word vectors. Vector-based models are widely used in NLP tasks like word sense disambiguation [McCarthy et al., 2004] and text segmentation [Choi et al., 2001]. Though their use is widespread, little attention has been paid to methods of constructing phrase or sentence meanings by combining these word vectors. After the excellent work of Mitchell and Lapata [2008]'s proposing the vector-based general compositional distributional semantics model, various models that used to compose different sentence parts to a single vector or tensor representation for a sentence or phrase are then well investigated. Central in these models is the concept of compositionality, which holds that the meaning of complex expressions is determined by the meanings of their components and the rules used to combine them. For this property of compositional models, it is very suitable to apply them on our aspectual classification problem for the aspectual class of a clause is also determined by the main verb and its complex interaction with the context.

2.2.1 General compositional models

The general compositional model proposed by Mitchell and Lapata [2008] defines the the composed result of two given vector representations, $\mathbf{v}$ and $\mathbf{u}$, as $\mathbf{p}$. The function

below describes the process:

$$p = f(u, v, R, K) \tag{2.1}$$

Above equation expresses the components that are used for constructing result vector. Apart from **u, v** and **p**, the **R** indicates syntactic relation that might have between these two words and the **K** stands for any additional information or knowledge that is required to construct the meaning or semantics. The function f represents the rule applied to combine these components. This compositional model is very general and can be extended to accommodate different needs by changing different required syntactic information **R**, additional knowledge **K** and modifying the combination rule.

Mitchell and Lapata [2008] then proposed two different compositional models based on the general one, an addition model and a multiplication model, to elaborate the extension of the general model. To simplify the situation, they ignore the **K** (additional knowledge) and fix the **R** by focusing on a well-defined specific linguistic structure (e.g. verb-subject structure). The additive model they propose is like:

$$p_i = u_i + v_i \tag{2.2}$$

This is achieved by modifying the function f to component-wise addition. This model is also called simple addition. And the simple multiplication model they extend is similar:

$$p_i = u_i \cdot v_i \tag{2.3}$$

All these two models hold the assumption of symmetry and thus are word order insensitive. To relax the symmetry assumption, one could apply weights to vectors and make them contribute differently such as $p_i = \alpha u_i + \beta v_i$, this is also called weighted addition model. In their experiments, they use bag of words (co-occurrence of neighbour text window) to represent each word vector. Since they ignore **K** and fix **R**, it makes these models hard to distinguish the differences between sentences with high lexical overlap.

2.2.2 Lexical function and Practical lexical function model

As discussed in above section, Mitchell and Lapata's models ignore syntactic information (**R**) and additional knowledge (**K**). However, from linguistic perspective, the semantics of a sentence should not just be a simple or weighted combination of words.

One component (e.g. an auxiliary) of a sentence should act as a function that affects other components (e.g. a verb). To bring in necessary linguistic information and interactions among different sentence components, the lexical function model (LF) was proposed [Baroni and Zamparelli, 2010, Coecke et al., 2010]. In the lexical function model, lexical vectors are arguments, functions that are used to take arguments are tensors (e.g. the determiner that combines with a noun). The order of a tensor (n+1) is determined by how many arguments (n) it could take. For instance, a noun is a vector or one-way tensor for it could not take any argument and it is an argument itself. Then an adjective is a two-way tensor or matrix, for it could take the noun it modifies as an argument. The result vector representation is calculated by vector and matrix multiplication. Baroni and Zamparelli's experiments only focus on adjective-noun (AN) structure (maximum two-way tensor) and their formula is like:

$$\boldsymbol{p} = \mathrm{B}\boldsymbol{v} \tag{2.4}$$

The p is the desired representation for adjective-noun phrase, and the v is the vector representation for noun. B is a matrix which refers to the representation for the adjective. The key point in this model is to estimate matrix representation for adjectives. Baroni and Zamparelli [2010] estimate the matrix representation for the adjective modifier in an effective way by linear regression. They first extract noun vectors and adjective-noun phrase vectors from corpus (from co-occurrence matrix). Then, the phrase vectors are regarded as labels for linear regression model to estimate the adjective matrix B together with input noun vectors. Since it uses the extracted phrase vectors as labels, it is an unsupervised method and therefore could save loads of annotation work and avoid complex iterative estimation procedure. Later, Grefenstette et al. [2013] extended their model to wider situations, from 2-way tensors to 3-way tensors. By extending to 3-ways tensors, it can explore more complex structures like transitive verbs (subject-verb-object).

Though the lexical function model captures interactions among different sentence components. It seems impractical to scale up to higher-order constructions because of the exponentially increased tensor dimension. For instance, if the vector representation for noun is of 300 dimension, then adjectives will be matrices of 300^2 cells and 300^3 dimensional tensors for transitive verbs. Modifiers of transitive verbs would have even greater representation size. This will lead to storage problem and make the learning process inefficient. Another issue is that same verb with different usage might be assigned with representations of different orders. For instance, the verb *eat* can be

used in both transitive and intransitive context (e.g. I eat an apple/I eat). Given this, the representation for transitive *eat* can not be used in an intransitive context and therefore need to train different representations for each possible usage that word might have. One problem arises when an usage has not been seen in the training data, it could not exploit other usage representations of the word to help it when the usage is presented in test data for the incompatible dimension. Secondly, it seems impossible to explore obvious relations among different usages of the same verb and between directly related words (e.g. demolish and demolition), since they have to be mapped into tensors of different orders and not directly comparable (e.g. the similarity between a vector and a matrix).

To handle above mentioned issues in the lexical function model, Paperno et al. [2014] proposes an improved lexical function model called practical lexical function (PLF). In the PLF model, a word is not represented by a single tensor of different orders. It is represented by a lexical vector plus an ordered set of matrices with each matrix standing for a function that takes specific argument. Each matrix is trained in the same way as in the lexical function model by linear or Ridge regression. When adding an argument, the model simply adds a matrix to the representation rather than resulting in a higher order tensor like in the lexical function model. A single representation for a sentence or phrase is obtained by applying matrices to corresponding argument vectors and summing over all resulting vectors. In PLF model, the general representation for a word is like:

$$< \vec{x}, \overset{\square_1}{x}, ..., \overset{\square_n}{x} > \tag{2.5}$$

It is an ordered tuple of a lexical vector and n matrices. The number of matrices in the representation indicates how many arguments the word could take. Normally, nouns only have lexical vectors, adjectives have one matrix apart from the corresponding lexical vector and transitive verbs might have two matrices, one for taking subject and another for taking object. By constructing such an expression, it is practical and easy to scale up to arbitrary long sentences without increasing the dimension exponentially. To correctly compose words together, the model incorporates two semantic composition rules. The first one is used to combine words with different arity, it is called *function application*. The process of composition is illustrated in Figure 2.1. If two words have different arity, the higher-ordered word is treated as the functor and the result is composed by multiplying the last matrix or the argument corresponding matrix in the functor by the argument lexical vector and summing the result to the functor.

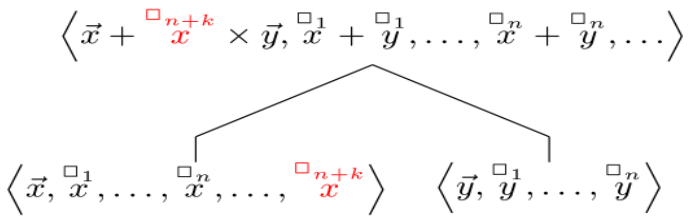

Figure 2.1: Function application: PLF composition rule for words with different arity.

For instance, word *cats* in the model has representation like $\overrightarrow{cats}$ because it is a noun and does not take any argument, the representation for it is just the corresponding lexical vector. This is same for *dogs* with representation as $\overrightarrow{dogs}$. The transitive verb *chases* in the model is represented as $< \overrightarrow{chases}, \overset{\square_s}{chases}, \overset{\square_o}{chases} >$. The $\overset{\square_s}{chases}$ stands for the matrix *chases* uses to take subject argument and the $\overset{\square_o}{chases}$ is used to take object argument. The composition process is illustrated in Figure 2.2.

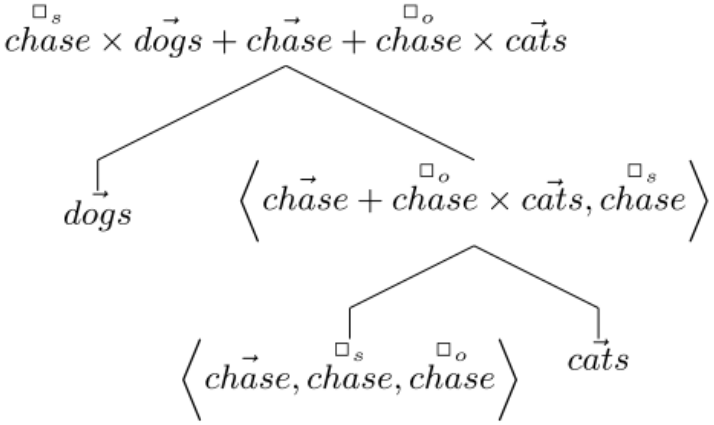

Figure 2.2: Apply function application twice to compose the representation of a transitive sentence.

As shown in Figure 2.2. The model firstly applies the rule to combine the verb *chases* with its subject noun *cats* by multiplying the subject matrix and the subject lexical vector. Then the resulted representation is applied again to obtain the final representation by multiplying the object matrix and the object lexical vector and finally summing up together.

The second composition rule is called *symmetric composition*, which is used to compose words of same arity (e.g. two vectors). In this situation, the model just sums two word representations pointwisely, vector with vector and n-th matrix with

n-th matrix. The PLF model treats most grammatical words including conjunctions as empty representation and does not project them into the semantics. Examples of the symmetric composition given in the original paper are also presented in Table 2.3:

sing: $\overrightarrow{sing}, \square sing$	dance: $\overrightarrow{dance}, \square dance$
sing and dance:	$\overrightarrow{sing}+\overrightarrow{dance}, \square sing+\square dance$
rice: $\overrightarrow{rice}$	cake: $\overrightarrow{cake}$
rice cake:	$\overrightarrow{rice}+\overrightarrow{cake}$

Table 2.3: Examples of symmetric composition rule

As mentioned before, grammatical words are treated as empty elements and this is why the word *and* is not added to the composition process in the second row in Table 2.3. Compared to the LF model, the PLF model does not have separate complete representation for each usage of a same word but to just add or remove functional matrices, therefore, all usages of the word are encoded in a single representation. By doing so, all different words are comparable for they all keep the basic lexical vectors. It is also able to handle usages that do not occur in the training data. For instance, if a verb used intransitively is encountered in test set while only occurred transitively in training set, we could leave the object matrix intact and only interact with the subject matrix and lexical vector, which is similar to a back-off strategy and is not considered by the LF model. On the other hand, if a verbs with more arguments is encountered in test data, the model would just add diagonal identity matrices to the representation tuple. Finally, unlike LF model's exponentially increasing dimension, the representation size of the PLF model now only increase linearly which makes the storage and parameter estimation more efficient.

The original PLF model has been analysed by Gupta et al. [2015] and they find an inconsistency in the model that the addition of the predicates lexical vector at test phase amounts to a systematic over-counting of the predicates lexical contribution. They thus propose a modified version of the PLF model by either remove the predicate lexical vector at test phase or adjust the loss function at training phase.

Another improved and simplified version of tensor-based model is proposed by Weber et al. [2017] which represents an event by 3-way tensor of fixed subject, predicate and object structure. To reduce the required number of parameters in LF, it chooses to train shared general tensors and then uses lexical vectors to scale those tensors to produce a predicate-specific tensor. This model is not as flexible as the PLF model and is

event-specific. Therefore, it is difficult to extend to clause level to accommodate other components like modifiers. Ritter et al. [2015] compares six different compositional models (including addition/multiplication, LF and the original PLF) in detail on its painstakingly constructed dataset in which a sentence's meaning cannot be determined by individual words to test the ability of composition on the task of described spatial relation classification. It surprisingly finds the effectiveness of addition for modelling complex interactions among different components compared to other complex models like PLF and LF. Buljan et al. [2018] also explore the ability of composition of three different compositional distributional semantics models (addition/multiplication and two versions of PLF) and other dedicated task specific models on a lexical substitution task with fixed adjective-noun-verb-adjective-noun (ANVAN) sentence structure and find that the PLF seems to be the best performing compositional model.

According to previous work stated above, the addition model is simple but effective. The PLF model is very flexible and a good choice when handling complete or arbitrary long sentences. Therefore, in our project, the simple addition model [Mitchell and Lapata, 2008] and the practical lexical function model [Paperno et al., 2014] are applied to the aspectual classification problem to assess their abilities of composition and to find the most contributing part by combining different sentence components.

2.3 Deduplication process to a Corpus for Aspectual type classification

Many applications (e.g. recommender system) taking advantages of natural language processing (NLP) techniques are facing the problem of detecting which documents are duplicates of each other and the tricky problem begins even before the real processing part [Ancin et al., 2012]. Because many NLP tasks tend to be computationally expensive, it is desirable for them to only apply these processes to new incoming documents. It is also very important to keep duplicates out of training data to prevent from biasing the trained model. Moreover, storing duplicates in your database would be a waste of resources and unreasonably expand the dataset size which might lead to computational difficulties and memory errors.

The deduplication process generally is about finding exact duplicates (exactly the same copies) and near-duplicates (with similar patterns to each other). The definition of near-duplicate varies across different tasks. It could be defined as documents with

high lexical overlap (e.g. same article from different channels with minor changes) and also as documents with similar syntactic or semantic patterns. The demand for deduplication process on our corpus (the MIT recipe corpus) for the aspectual classification task is really strong. The MIT recipe dataset is made of cooking instructions, and, while the recipes themselves are not duplicates, they contain a large number of specific instructions that are syntactically or semantically similar. For example, the instruction *preheat oven to 150 degree* makes no difference with the instruction *preheat the oven to 300 degree* in terms of aspectual property, and the instruction *Shape a dough into a plait* has the same aspectual type with *Form the pastry into a ball*. Including all those similar instructions in the dataset would make the following work hard to do. Firstly, the MIT recipe corpus has not been labelled according to aspectual class which means that the annotation work is necessary. However, labelling each separate instruction in the dataset would require the annotator to repeat the same work for the instructions manifest the same patterns and could be annotated together in one shot. Therefore, it is necessary to identify what sentences do not have to be annotated manually. Secondly, by including these duplicated instructions, the trained classifier will be highly biased for certain instruction patterns. As a result, performing near-duplicates detection can significantly reduce the size of corpus and relief the required work for annotation, and also remove those potential bias impacts that might have on the training process. Due to the time limit, the project only investigates the deduplication pipeline on MIT recipe corpus and leaves the following annotation and classification problem to future work. The aspectual classification problem is mainly investigated on the other two datasets.

In following sections, two major techniques the project employed to detect near-duplicates will be described in detail.

2.3.1 Jaccard similarity and locality sensitive hashing

The first near-duplicate example we make in above section in the MIT recipe data is *preheat oven to 300 degree* and *preheat oven to 150 degree*. In the recipe corpus, though each recipe is being different, they still have standard things to do and standard ways of describing them such as the *preheat* event can be a standard process in many recipes. Such standard processes would have high-level lexical overlap for they share the same action with only various degrees, tools, time spans or ingredients. One good way of finding similar instructions with lexical overlap is by comparing the jaccard

similarity. The jaccard similarity of two sets A and B is given below:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

The $J(A,B)$ is the ratio of the size of the intersection of A and B to the size of their union. To compare two documents by jaccard similarity, each document should be represented as a set. One could use the word sequence of a document as a set. But to allow slight textual variation, the most common and effective way of representing documents as sets for the purpose of identifying lexical similarity is to construct from the document the set of short strings that appear within it [Leskovec et al., 2014]. One implementation of it is called shingling. For instance, if 2-shingle or 2-char n-gram based shingle is applied to the phrase *Cats sing.*, it will result in a set like $\{'Ca','at','ts','s\ ','\ s','si',...,'g.'\}$. There are options how space symbols like blank, tab are treated. Normally, the blank is kept and a sequence of one or more blanks(e.g. tab) is turn to one blank. To better utilize the jaccard similarity, slightly various expressions (e.g. temperatures, times, amounts) should be normalized into the same token to obtain higher similarity score. This process will be described in section 4.2 in detail.

It seems that after applying shingle and jaccard similarity, we could already be able to find near-duplicates with lexical overlap. However, it is impractical for large corpus to directly use shingle and jaccard to search duplicates for finding those duplicates take quite a long time as computing the jaccard similarity of the documents requires comparing every document to every other document. But we do not want to compare every document with others, we just want to obtain document pairs that might have high jaccard similarity. The locality sensitive hashing with minihash is brought in to handle this situation.

Document Shingles			
row	shingle ID	Doc 1	Doc 2
1	1	0	1
2	2	1	1
3	3	0	1
4	4	1	0
5	5	1	1
6	6	0	0

Table 2.4: Example 1 for minihash

Document Shingles			
row	shingle ID	Doc 1	Doc 2
1	6	0	0
2	2	1	1
3	3	0	1
4	1	0	1
5	4	1	0
6	5	1	1

Table 2.5: Example 2 for minihash

Locality sensitive hashing (LSH) is a hashing method that attempts to keep the local relation of the data while significantly reducing the dimension of the dataset [Indyk and Motwani, 1998]. It is a slightly different hashing technique compared to other hashing methods as it tries to ensure hash collisions, which is something traditional hashing techniques try to avoid, for similar items. The general aim is to reduce the number of comparisons needed to find similar items, the hash collisions come in handy here as similar documents have a high probability of having the same hash value. The result hash value then is treated as an ID or address for a bucket that contains potential similar items. This allows one to just compare items in the same bucket rather than comparing every item to every other item in the dataset. However, each document might has a different number of shingles regarding the length of the document. So, a fixed length representation should be created before applying LSH to documents. The LSH actually relies on two methods, the first one is to turn a document to a fixed length hash fingerprint. The second one is to apply LSH on that fingerprint to put it into the bucket. The fingerprint creating process is done by the minihash algorithm. Consider the two document shingles shown in Table 2.4 (An entry of zero (0) indicates that a particular shingle does not appear in the document, while an entry of one (1) indicates that is does occur.). Given these entries, the jaccard similarity of the two documents is $\frac{2}{5}$. A minihash is just a random permutation function of the rows and gives back the row index where the first non-zero entry is located. Table 2.4 and Table 2.5 represent two different row permutation and thus get different minihash result. In Table 2.4, the minihash for Doc1 is 2 and 1 for Doc2 which indicates that they are not similar, while in Table 2.5 the minihash is 2 for Doc1 and also 2 for Doc2 which means that they are similar. By conducting permutation multiple times, for instance 100 times, we will get 100 minihashes for each document. The 100 minihashes together are called a hash fingerprint for a document. The minihash has the property that the probability of a hash collision for a minhash is exactly the jaccard similarity of two sets. By increasing the number of minihashes in a hash fingerprint, the closer the jaccard similarity result it will get compared to the original shingled text.

After obtaining the hash fingerprints, the LSH will be applied to it. The idea of LSH is just to divide the fingerprint into several pieces, and then use another hash function (not minihash) to hash those pieces to get bin IDs for each piece and put the whole fingerprint into each bin that each piece happens to land in. For example, if a fingerprint with 100 minihashes is divided into 10 pieces, each piece will contain 10 minihashes. A hash function (not minihash) is then applied to those pieces to get 10

bin IDs. Then, the whole fingerprint is stored in every bin that each piece happens to land. The dividing into pieces process is the locality in locality sensitive hashing, and the hashing is the hash function that maps pieces to bin IDs. After processing the whole dataset, all item pairs hashed into the same bin are called candidate similar pairs and thus we only have to compute jaccard similarities for those candidate pairs rather than the whole dataset. Since not all documents will land in the same bins, we have significantly reduced the comparisons.

The jaccard similarity and LSH method are mainly used to detect the duplicates of high lexical overlap (the first duplicate example made at the beginning of the section). If the detection of other kinds of duplicates are in need, other methods should be taken advantage of.

2.3.2 Latent semantic Analysis

After applying LSH to the MIT recipe corpus, near-duplicate bins or clusters are found to cover around 30% to 40% instructions depending on the corpus size. These kinds of near-duplicates are mainly of lexical overlap, by detecting them it handles the first type of near-duplicate as mentioned in above section. The second type gives in the beginning is the *Form the pastry into a ball* and *Shape a dough into a plait*. This kinds of near-duplicate pairs are more than simple lexical overlap. They demonstrate both syntactic and semantic similarity. With little lexical overlap, the LSH and jaccard similarity fail to detect them. Therefore, we need to find another way to obtain near-duplicates that are syntactically and semantically similar. The latent semantic analysis (LSA) is a common and effective way of finding the hidden relation behind documents and terms they contain by producing related concepts or topics [Deerwester et al., 1990] and often be used as a indexing method (so it is also called latent semantic index). Thus, it is quite suitable in our situation to detect syntactically and semantically similar near-duplicates based on our already detected bins.

The LSA model assumes that two words are close in meaning if they occur in similar text. For instance, if the word *cats* frequently co-occurs with *dogs*, then a document talks about dogs is considered to be more similar to a document about cats than to a document about *food*. The purpose of the LSA model is to find out the actual meaning of a term in documents and queries, which is the latent semantics. Technically speaking, it aims to use lower dimensional space to model a large collection of documents and project all terms and documents to that space before comparing their relations

[Landauer et al., 1998]. For instance, if a dataset has 1000 documents and a vocabulary of 5000 words, one could construct a term-document matrix of size 5000×1000. Each row in the matrix represents a word and each column a document, the value of the matrix cell(i,j) indicates the number of occurrence of word i in document j. The process of applying LSA on this term-document matrix is the process of conducting singular value decomposition (SVD), dimension reduction and the construction of latent semantics space. The formula for SVD process is as below:

$$X = U\Sigma V^\top \tag{2.7}$$

As shown in equation 2.7 and Figure 2.3, X is a $M \times N$ matrix, the U has size of $M \times R$, Σ is a $R \times R$ diagonal matrix, and the V is a $R \times N$ matrix. The rows of $V^\top$ contain eigenvectors of $X^\top X$. The columns of U contain eigenvectors of $XX^\top$.

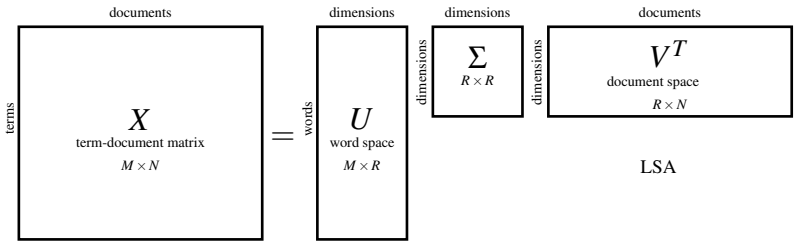

Figure 2.3: Apply SVD on term-document matrix representation

After performing the SVD process on the original term-document matrix, the LSA model then only keeps the largest k singular values in the Σ matrix and set following to 0 to construct a new low-rank Σ_k matrix where k is often set to be 100 to 300 dimensions. By keeping only largest k entries of Σ, we could represent the new U as U_k of size $M \times k$ and the $V^\top$ as $V_k^\top$ of size $k \times N$. The SVD operation, together with this reduction process, has the effect of preserving the most important semantic information in the text while reducing noise and other undesirable effects of the original space of X. After the reduction, the X_k will be reconstructed:

$$X \approx X_k = U_k \Sigma_k V_k^\top \tag{2.8}$$

The reconstructed X_k is the representation of X in latent semantics space. The LSA model is often used as an indexing and query tool. The process is like:

$$V_q = X_q^\top U \Sigma^{-1} \tag{2.9}$$

Firstly, the query document is represented as a vector of words. Thus, it could be regarded as the q-th column of the X matrix which is the X_q as shown in equation 2.9.

By following above equation, the new vector V_q is created for a query or for a new document. After that, the cosine similarity or other metrics between the query and all other documents could be calculated iteratively.

Our finding more near-duplicates task is actually a query task. The results of LSH are instruction clusters with each cluster a group of near-duplicates. To find more duplicates based on the LSH clusters, we could build a LSA space from the entire MIT recipe corpus and use clusters as queries to query or rank instructions in the corpus and returns most similar instructions to each cluster. By dong so, we could add the most N similar instruction we get into the existing clusters.

Chapter 3

Aspectual type classification

In this chapter, two classification experiments in terms of different apsectual distinction will be presented. The section 3.1 would mainly focus on the distinction between state and event (achievement, accomplishment, activity). Experiments and result analysis on the distinction between telic (accomplishment, achievement) and atelic events (activity) are described in section 3.2.

Following previous work on the aspectual classification problem [Siegel and McKeown, 2000, Friedrich et al., 2016, Friedrich and Palmer, 2014], the problem in this project is still regarded as a supervised classification task. The classifier used in all experiments of both distinctions is the Support Vector Machine (SVM) with default parameter settings implemented in Scikit-learn [Pedregosa et al., 2011]. Before feeding individual features to the classifier, all features are composed together by compositional distributional semantics models (CDSM) as described in Chapter 2. Two CDSMs are investigated in this project, the first is the simple addition model and the second is the original PLF model. Unlike previous work, we do not take use of any linguistic indicators, we only employ the distributional word vectors as features.

3.1 Event/State distinction

3.1.1 Dataset

The event/state distinction experiments are conducted on the Situation Entities (SitEnt) dataset [Friedrich et al., 2016]. The SitEnt dataset, which is also called MASC-Wikipedia corpus, consists of approximately 40,000 multi-genre clauses. Those clauses are extracted from MASC [Ide et al., 2010] and Wikipedia documents by using SPADE

[Soricut and Marcu, 2003] with some heuristic post-processing [Friedrich et al., 2016]. Each clause is then annotated by three annotators in terms of the seven aspectual types of a clause (state, event, report, generic, generalizing, question, imperative). The results of three annotators are merged into a gold standard by majority voting. Annotators are not forced to label every sentence, so the category of a sentence could be *none*. Since we only care about the event and state distinction, other classes like report and question are ignored. By extracting event and state sentences from the original dataset, we form a new dataset with class distribution shown in Table 3.1. The new State/Event dataset is 2-class problem with each clause being labelled as state or event. It is an imbalanced dataset with approximate 2:1 distribution. We follow the original train/test split strategy as indicated in the dataset field column.

	State	Event
Train	14793	7955
Test	3544	1733

Table 3.1: State/Event Class distribution for extracted SitEnt dataset

3.1.2 Preliminary

Since the aspectual class of a clause is mainly determined by the property of the main verb and its interactions with the context, we construct input features by investigating those complex interaction relations. The direct representations of these relations could be thought as the result of dependency parsing. For example, interactions between the main verb and different sentence components for the sentence *You can have a life outside your computer* are clearly presented in Figure 3.1. The main verb here is the root *have*, interactions that take place in this sentences with the main verbs are all directly pointed with arrows, specific relations are labelled on the edge.

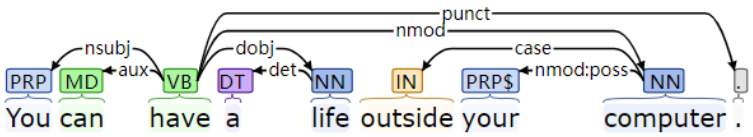

Figure 3.1: The result of stanford dependency parsing on the sentence *You can have a life outside your computer*

The main verb has five direct relations or connections. The first one is the *can* which is marked as **aux** (auxiliary). Then the *You* is the noun subject (**nsubj**) of the

verb while the *life* is the direct object **dobj** of the verb. This is tha same for the noun modifier (**nmod**) *computer* and the punctuation (**punct**) for the verb. We take those words that are directly related to the main verb as features to express these interactions. By doing so, we make the assumption that the aspectual class of a clause can be captured by the combination of the main verb and its directly related sentence components' distributional semantics representations. So, firstly we conduct standard stanford dependency parsing [Klein and Manning, 2003] on the whole dataset to obtain dependency relations for every clause. Then, we deliberately select words that are directly related to the main verb, these words are also called the first order children of the main verb. These picked out words then are regarded as potential features used for the classifier. As the SitEnt dataset indicates the main verb for each clause, we do not need to worry about extracting them. We also gives the 10 most common dependency relations a main verb might have across the SitEnt dataset in Figure 3.2.

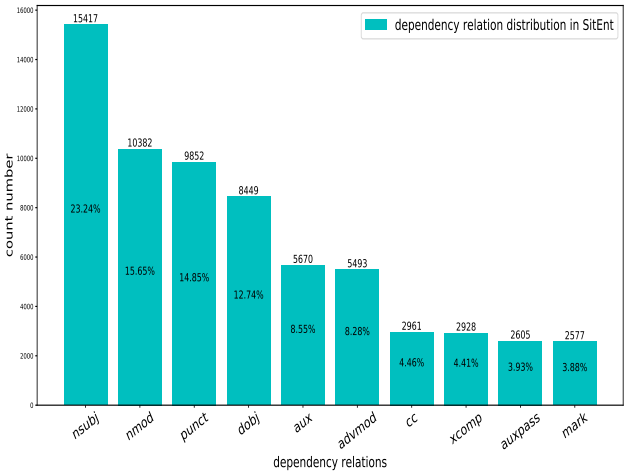

Figure 3.2: The 10 most common dependency relations for the SitEnt dataset

After constructing those potential features, we now move to the step of representing them. Since we are exploring the CDSM, the word embedding comes as the first choice. Due to the training process limit of the PLF model, we choose to train our own word embedding on the SitEnt dataset. All words are assigned a 300-dimensional vector which is obtained by utilizing the skip-gram word2vec model implemented in gensim [Mikolov et al., 2013, Řehůřek and Sojka, 2010]. Apart from word vectors, the training of PLF model requires phrase vectors for different kinds of relations. To obtain desired phrase vectors, we firstly combine two words in a clause with specific relations to a single token (e.g. turn phrase *could do* to *could_do*). Then, potential contexts

are put right after the combined token in case there are other words between the two target words. For instance the relation *nsubj* might be detected in a sentence like ***I*** *fully* ***understand*** *it*, to combine the target words into a single token, the information in between should also be preserved. The result sentence will be like *I_understand fully it*. By feeding those combined data to the word2vec model, the phrase vector for different relations could be obtained. After obtaining phrase vectors, we could use both word vectors and phrase vectors to get specific functional matrices for the PLF model in the way introduced in Chapter 2. This is achieved by exploiting the *DISSECT* toolkit [Dinu et al., 2013] with Rige regression.

3.1.3 Experiment methods

We investigate two different compositional models on the classification problem, the first one is the simple addition model and the second one is the practical lexical function model. The simple addition model follows below process:

$$S = v + \sum_{c \in C} c \tag{3.1}$$

The v is the vector representation for the main verb and C are all first order children of the verb (directly related words). We could choose which child to be combined into the representation. The S is the resulting representation for the clause which is then used to feed the SVM classifier to predict the aspectual class of the clause.

The practical lexical function model is not just a simple combination. After the preliminary process, each word in the PLF model is then represented as:

$$< \vec{x}, \overset{\square_1}{x}, ..., \overset{\square_n}{x} > \tag{3.2}$$

When composing the representation of a clause, it follows the function application and symmetric composition rules as described in Chapter 2, which are mainly vector and matrix multiplication and summation over results.

We firstly use all first order children of the main verb and the verb as features to experiment with. Then, we conduct a series of experiments by adding or removing specific children to explore which child word contributes the most to the determination of the aspectual class of the given clause. These feature selections and combinations are all investigated on both two compositional models and the selection strategy is based on the linguistic consideration as well as the relation frequency as shown in Figure 3.2. Thus, we choose *nsubj*, *dobj*, *aux* and the *negation* to experiment with.

3.1.4 Results and analysis

Experiment results are all shown in Table 3.2. In the section of addition models, the best result is obtained by the model with all children plus verb as features. This model is also the best performed model if considering overall. The worst model is the children-only model or also the verb-only model if considering the average F1-score of two classes. Though being the worst, the verb-only have gained competitive results with the model considering all children. This might indicate that the distributional semantic representation of the verb is the most informative and important part compared to other components when determining the aspectual class of the clause in this dataset.

Model	F1-score(state)	F1-score(event)
Addition model(all children plus verb)*	**0.90**	**0.80**
Addition model(verb-only)	0.85	0.71
Addition model(all children-only)*	0.86	0.70
Addition model(verb and aux)	0.89	0.79
Addition model(verb, aux and negation)	0.89	0.79
Addition model(verb, aux, nsubj)	0.88	0.78
Addition model(verb, aux, nsubj, dobj)	0.89	0.77
PLF model(verb-only)	0.85	0.71
PLF model(verb and aux)	0.88	0.78
PLF model(verb, aux and negation)	0.88	0.78
PLF model(verb, aux, nsubj)	0.88	0.77
PLF model(verb, aux, nsubj, dobj)	0.88	0.76
Friedrich et al. [2016]'s model	0.80	0.78

Table 3.2: Experiment results on addition and PLF models with different sentence components. The model with a star on upper right means this feature set is only applied on this model

Starting from the verb-only addition model, we obtain the second best model by adding *aux* words. The results of this model (with only verb and aux features) are only 1% lower than the best model with all features. According to this comparison, the auxiliary in a clause seems to play a crucial part when determining the distinction between event and state. When we continue adding the negation words into composition, the result seems to remain stable which might indicates their irrelevance when considering

the classification. Following that, we can see from the table that the performance of the addition model continues to drop when *nsubj* and *dobj* related words are included to the composition. This phenomenon shows that subjects and objects in the SitEnt dataset might contain noises and therefore fool the classifier to make wrong decisions. From above comparisons, it is clear that the property of the verb which is encoded in corresponding distributional semantic representation is the most informative part in determining whether a clause is a state or event. Another reason for the good performance of the verb might be that verbs in the SitEnt dataset are mostly unambiguous in terms of their aspectual classes, only 920 out of $\approx$ 4.5k verb types in the training set occur with both labels [Kober et al., 2018]. Therefore, one model could distinguish event and state well simply according to the distinct verb. This is something that needs to be handled in the future to make sure that it is the verb and the context it occurs in together make the decision. Among the related words with the main verb, the auxiliary contributes the most to the classification and itself makes the classification result competitive to the best model. The finding of the importance of auxiliaries fills our expectation, for the distinction is made between event and state. As described in Chapter 2, the activity and accomplishment are often combined with progressive tense while the state is not natural to do so. Moens and Steedman [1988] also indicate that the aspect including the progressive and the perfect could force one aspectual class to transit to another. The progressive and the perfect are all led by auxiliaries and therefore, we would expect the auxiliary to contribute a lot. Some example sentences which are correctly classified by involving *aux* are presented in Table 3.3. These examples show how the aspectual transformation takes place.

Sentences	Verb-only model result	plus-aux model result	Correct Label
The first responsibility should be driving safely	event	state	state
The author had died	event	state	state
It's about building community	event	state	state

Table 3.3: Example sentences that are correctly classified by adding *aux*

Similar result trending could be observed in the PLF model section as shown in Table 3.2. Still the model with verb plus aux features hits the best F1-score for both classes. Then by adding *nsubj* and *dobj* words, the performance of the model keeps dropping. The results of PLF model though being slightly worse than addition model, they still confirm our previous expectation and analysis. The PLF model normally needs large amounts of training data (usually trained on the concatenation of ukWaC,

Wikipedia and British National Corpus) for the estimation of well represented phrase vectors and functional matrices. In this experiment, we only use the SitEnt dataset (less than 50,000 sentences) for training these vectors and matrices. We do not expect the PLF model to be well-performed though it produces competitive results compared to the addition model. The simple addition model is still extremely effective, although it ignores the word order information.

Our vector-based compositional models with *aux* and verbs outperform Friedrich et al. [2016]'s model which uses a large number of linguistic features for classification. This indicates the compositional models' ability of capturing semantic interactions in a clause. Also, it reflects the ability of the distributional representation for representing both syntactic and semantic meanings.

3.2 Telic/Atelic distinction

3.2.1 Dataset

The dataset used for telic and atelic distinction is derived from the telicity dataset proposed by Friedrich and Gateva [2017]. The original dataset consists of 1863 sentences extracted from 10 English texts from the MASC corpus and labelled as either telic or atelic. The original dataset is highly imbalanced, with 82% sentences labelled as telic. Training the classifier on such a highly imbalanced dataset makes the model biased to the class of larger proportion. Another issue about the dataset is that only 70 out of about 570 distinct verb lemmas occur with both labels [Friedrich and Gateva, 2017] and if looking at sentences, only 24% atelic sentences have verbs occur in telic context. Being like that, a verb-only model described in above section could have strong performance on the classification task. To address above two issues, we plan to add more atelic sentences with verbs already occur in the telicity dataset in telic context.

New potential atelic sentences are extracted from the Stanford Natural Language Inference (SNLI) dataset [Bowman et al., 2015] and then are selected and labelled by Professor Bonnie Webber. 396 new atelic sentences are added into the original telicity dataset, which forms a dataset with class distribution presented in Table 3.4. Now, the new dataset is more balanced with 67% sentences labelled as telic. More importantly, the new dataset currently have 472 out of 733 (64%) atelic sentences with verbs occur in telic context. By doing so, verbs in the telicity dataset now could be considered to be ambiguous and therefore, a verb-only model is assumed not to be able to make well

	Telic	Atelic
Train	1214	586
Test	304	147

Table 3.4: Telic/Atelic Class distribution for new constructed telicity dataset

classification results and the impact of context can be emphasised.

3.2.2 Preliminary

The preprocessing work for telic/atelic task is basically the same as above event/state task. Firstly, we apply stanford dependency parsing on the whole dataset to extract relations and find words that directly related with the main verb. The distribution of dependency relations in the telicity dataset is quite similar to that in the SitEnt dataset so we do not display it again. The word representation process is slight different from previous one. Due to some writing and storage limitation of the DISSECT toolkit [Dinu et al., 2013], we choose to represent words with 100 dimensional vectors rather than 300 dimensional vectors to make it able to write to file. Then we follow the same pipeline of obtaining specific phrase vectors and functional matrices.

3.2.3 Experiment methods and additional replacement

The classifier and compositional models used in this task is exactly the same with previous task. We choose *nsubj*, *dobj* and *aux* to experiment with. We firstly try experiments on simple addition model.

Model	**F1-score(telic)**	**F1-score(atelic)**
Addition model(all children plus verb)	0.89	0.69
Addition model(verb-only)	0.84	0.52
Addition model(all children-only)	0.89	0.68
Addition model(verb, aux and negation)	0.85	0.52
Addition model(verb, aux, nsubj)	**0.89**	**0.70**
Addition model(verb, aux, dobj)	0.85	0.54

Table 3.5: Experiment results of addition model on telicity dataset

According to above results, the verb-only model has the worst performance on de-

ciding the aspectual types. This is due to our efforts of adding sentences with verbs in telic context, making verbs in the dataset ambiguous to reflect the impact of context. By several comparisons, we find that the *nsubj* seems to contribute the most to the classification. This phenomenon is quite strange as the subject is not expected to be the most informative indicator for aspectual class from linguistic perspective. By checking sentences in the telicity dataset, we find that the reason might be the unambiguous noun subjects. New adding atelic sentences are extracted from the SNLI dataset which is based on image captions. A typical sentence in it is like *A man is standing in front of a car*. Therefore, subjects like *man*, *woman*, *people* are more common in the atelic sentences and thus make the distinction more obvious. After counting those typical subjects, almost 90% of them occur in atelic context. To relax this problem, we plan to replace all subjects across the data to specific placeholders. We experiment with two different strategies. The first one is to replace all subjects with a unique placeholder *proall*, by doing so, we do not distinguish the number of the subject. The second strategy is to replace subjects with *prosg* if the original subject is singular and *propl* if it is plural. Replacing all subjects with those placeholders, we assume that the only contribution the subject might be able to make considering the telic/atelic distinction is its number. So we experiment with these two replacement strategies and the related features as before. After replacing subjects, all word vectors, phrase vectors and functional matrices need to be retrained on the replaced version of data.

3.2.4 Results and analysis

Experiment results for simple addition model with different subject replacement strategies are presented in Table 3.6. For both two different strategies, the verb-only model has the worst performance while the all-feature model performs the best. This is due to the large number of ambiguous verbs in the dataset, the verb-model could not well distinguish them. When making the distinction between telic and atelic in replace-proall models, the *aux* remains the most contributing part. Though being replaced with the same token, the subject still has an impact on the determination of aspectual class. However, the pattern changes when all subjects are replaced by prosg or propl. In the prosg/propl model, the *aux* now becomes a trivial part by improving only 1% F1-score based on verb-only model. Currently, the most contributing part is the *nsubj*. Recall our previous assumption that the only contribution a subject could make to the decision of aspectual class is its number. By comparing those two strategies, we could say that

Model(replace with proall)	**F1-score(telic)**	**F1-score(atelic)**
Addition model(all children plus verb)	**0.89**	**0.69**
Addition model(verb-only)	0.83	0.51
Addition model(all children-only)	0.87	0.58
Addition model(verb, aux)	0.85	0.55
Addition model(verb, aux, nsubj)	0.85	0.57
Addition model(verb, aux, dobj)	0.85	0.56
Addition model(verb, aux, dobj, nsubj)	0.85	0.55
Model(replace with prosg/propl)		
Addition model(all children plus verb)	**0.89**	**0.71**
Addition model(verb-only)	0.83	0.57
Addition model(all children-only)	0.87	0.62
Addition model(verb, aux)	0.85	0.58
Addition model(verb, aux, nsubj)	0.86	0.64
Addition model(verb, aux, dobj)	0.86	0.62
Addition model(verb, aux, nsubj, dobj)	0.87	0.67

Table 3.6: Experiment results on addition models with different sentence components and with different replacement strategies. Bold figures indicate the best model in its model section

the assumption might be correct, the number of subject does have an impact on the classification decision. Then by adding the *dobj*, the improvement is significant while the *dobj* only has limited influence if looking back to the proall model. We expect to see the direct object plays an important role in determining the classification in both replacement strategies while the subject might only have limited impact. However, both experiment cases give us surprising results. Some example sentences that are correctly classified by adding *nsubj* and *dobj* are shown in Table 3.7.

Sentences	Verb-only model result	plus-nsubj&dobj model result	Correct Label
Several propl put clothes on their heads	telic	atelic	atelic
Some propl took a stroll in the autumn afternoon	telic	atelic	atelic
The propl write letters to their loved ones	telic	atelic	atelic

Table 3.7: Example sentences that are correctly classified by adding *nsubj* and *dobj*

Another problem about the result is that the verb-only model has different results for different replacement strategies. Since the model only contains the verb and the verb is not replaced with other tokens. The result for these two verb-only model

Model(replace with proall)	**F1-score(telic)**	**F1-score(atelic)**
PLF model(verb-only)	0.83	0.51
PLF model(verb, aux)	0.83	0.55
PLF model(verb, aux, nsubj)	0.84	0.58
PLF model(verb, aux, dobj)	0.84	0.55
PLF model(verb, aux, dobj, nsubj)	0.85	0.60
Model(replace with prosg/propl)		
PLF model(verb-only)	0.83	0.57
PLF model(verb, aux)	0.83	0.56
PLF model(verb, aux, nsubj)	0.85	0.59
PLF model(verb, aux, dobj)	0.83	0.59
PLF model(verb, aux, nsubj, dobj)	**0.85**	**0.61**

Table 3.8: Experiment results on addition models with different sentence components and with different replacement strategies.

should be exactly the same or at least similar. However, a big difference is found between them. The reason can only be the difference between two versions of the word embedding. By replacing all subjects to proall or prosg/propl, skip-gram models trained on these two different datasets give highly different results. Our assumption is that by replacing all subjects with different tokens, the representation of a verb now is learnt from different contexts which makes the representation differ substantially from what we had before. Furthermore, there is a fair amount of stochasticity involved in learning word2vec representations from a relatively small corpus such as the SitEnt dataset [Antoniak and Mimno, 2018]. These two or more factors together make the variation of embeddings for verbs. The other issue is the subject problem. We find that by replacing them with prosg/propl, the subject is contributing more compared to proall replacement. Apart from our assumption of the number of the subject, more experiments should be done to investigate what kinds of subject pattern get improved. Due to the time limit, these work will be left to the future.

Then we continue our experiments on the PLF model with exactly same settings, and the results are shown in Table 3.8. Similar to the event/state experiments, the PLF model gives results of similar trending or patterns to the addition model. For instance, *nsubj* in the prosg/propl model contributes more to the decision of classification than in the proall model. The *dobj* helps to improve more in the prosg/propl model while it

has limited impact in the proall model. Though PLF-based models have worse performances compared to simple addition model, the same pattern indicates the correctness of its compositionality and reflects the basic relationship among different sentence components. Considering the limited data for training such complex PLF models, it is still worth exploring its ability of semantic composition.

Chapter 4

Corpus deduplication

In this chapter, the deduplication process pipeline on the MIT recipe data will be introduced in detail. In section 4.1, the dataset is described as well as its patterns and some necessary preprocessing steps are given in section 4.2. In section 4.3, we explain our experiment methods in detail, concerning the stability of the method when handling with various sizes of the corpus. Finally, the results are reported in section 4.4

4.1 Dataset

The deduplication process is done on the MIT recipe corpus [Salvador et al., 2017]. It consists of over one million recipes, and each recipe contains ingredients and step-by-step instructions with each step a sentence. By collecting those instructions, the MIT recipe corpus results in more than 10 million instructions. A large number of these instructions are imperative and incomplete, dropping information that encoded in contexts. Being mostly imperative, this dataset is different from the SitEnt and the telicity dataset, since the latter two are mainly collected from news, blogs and journal articles. This corpus is also unlabelled. To make use of it for the aspectual classification, annotation needs to be done. As described in section 2.3, the large number of syntactically and semantically similar instructions makes the deduplication process necessary for saving the annotator's work and training a less biased classifier.

4.2 Preliminary

The LSH method is mainly used to detect instructions of high lexical overlap. While in the recipe corpus, variations of a same pattern are common and we do not want to miss

any similar instruction due to the slight variation. For instance, the instruction like *preheat oven to 150 degree* and the instruction *preheat oven to 295 Fahrenheit* are definitely near-duplicates from our perspective in terms of apsectual type. However, the variational expression for temperatures makes significant difference when calculating the shingled jaccard similarity. The jaccard similarity for those two instructions after 3-char n-gram shingling is only 39%, far from being able to claim near-duplicates. If we replace these various temperature expressions to a specific placeholder like *TEMP*, the result jaccard similarity is improved to 100% with instructions now look like *preheat oven to TEMP*. This phenomenon brings in the need of normalizing phrases like temperatures. To detect more potential near-duplicates and relax the impact of variational expressions, we normalize all temperature expressions (number + temperature units like C, degree, d, F) with *TEMP*, time expressions (number + time units like day, seconds, hours) with *TIMEP* and measurement expressions (number + measurement units like tablespoon, cup, inch, tbsp) with *MEASP*. Apart from normalizing these expressions, we also combine ingredient conjunctions. For example, the instruction *mix pepper and garlic* is considered to be a near-duplicate of *mix water and baking powder*. These ingredient nouns are various and hard to match. We directly detect the *NP and |or NP* pattern (with several variations) after conducting POS tagging (still use stanford NLP toolkit) on the corpus and then replace these patterns with *CONJP*. Since the POS tagger is not perfect and makes errors from time to time, not all desired patterns are detected and replaced. To detect more near-duplicate instructions and better investigate the instruction patterns that make up a corpus, we split all conjunctions between actions into isolated instructions. For example, the instruction *Adjust oven rack to upper-middle position, and heat oven to 350 degrees* has two actions and when clustered by LSH we often see it being clustered into the *heat* cluster as well as the *ajust position* cluster which makes each cluster noisier and make the annotation work trickier. By splitting these conjunctions, we are able to explore the basic patterns that could make up a recipe corpus and get a cleaner cluster result. The annotator could just label each cluster rather than all instructions and give some combination rules among clusters with different labels. This process could save a lot of time for annotators.

4.3 Experiment methods

We firstly take a small subset of the whole 1 million recipe dataset to experiment with and then gradually increase the dataset size to evaluate the stability of the deduplication

if only apply LSH					
Subset size	number of clusters	instructions in clusters account for of the whole subset	type/token ratio for instructions in clusters	how many clusters are needed to cover 80% instructions in clusters	how many clusters are needed to cover 90% instructions in clusters
60K	3520	32%	78.38%	1405 (first 39.9%)	2402 (first 68.24%)
100K	6353	35%	77.38%	2228 (first 35.07%)	4075 (first 64.14%)
160K	9402	38%	76.74%	2916 (first 31.01%)	5741 (first 61.06%)
200K	11524	39%	76.20%	3364 (first 29.19%)	6885 (first 59.74%)
300K	16726	41.3%	75.48%	4290 (first 25.65%)	9417 (first 56.30%)
after applying LSA to the result of LSH					
60K	3520	69%	11.15%	1247 (first 35.4%)	2101 (first 59.69%)
100K	6353	75%	7.31%	1887 (first 29.7%)	3481 (first 54.79%)
160K	9402	78%	5.46%	2467 (first 26.24%)	4822 (first 51.29%)
200K	11524	79.8%	4.66%	2802 (first 24.31%)	5446 (first 47.26%)
300K	16726	82.4%	3.44%	3469 (first 20.74%)	7487 (first 44.76%)

Table 4.1: Experiment results after applying LSH with or without LSA on different subset size of the recipe dataset

process. Dataset sizes in $\{60K, 100K, 160K, 200K, 300K\}$ are investigated. Due to the limited computing resources, larger sizes are not explored. The LSH is firstly applied to dataset and result in a list of clusters with each consisting of near-duplicates with same pattern. Then the LSA is performed on the dataset and those clusters. Cosine similarities are calculated among them to add more semantically similar instructions into existing clusters (no new cluster added). We compare the same pipeline process on all five different subsets. The LSH we use is set to 4-char n-gram shingles and have fixed 100 minihash functions. We transform the term-document matrix to TF-IDF representation before applying LSA and the topic dimension for the LSA model is set to be 150.

4.4 Results and analysis

Experiment results are presented in Table 4.1. The third column of the table represents the proportion those instructions included in clusters account for of the whole instructions in the corresponding subset. The fifth and sixth columns represent that, if clusters are ordered in terms of the number of instructions in them, how many clusters are needed to cover a certain percentage of instructions included in clusters counting from the first cluster. A trending is found in both these two models (with or without LSA). By increasing the corpus size, more clusters are formed by LSH and account for an increasing percentage of the whole subset. The percentage of clusters needed to

cover 80% and 90% instructions in clusters is decreasing. The type in the type/token ratio metric refers to the distinct instruction. It is clear that as the corpus size increasing, the type/token ratio for instructions is dropping. This decline indicates that as more and more instructions included in the subset, the number of distinct instructions remains stable or increases slower than new instruction types. We could assume that the recipe instructions are probably zipfian, with fewer and fewer new or unseen instructions being found. When we look at clusters given by LSH, taking 100K, 200K and 300K's results as examples, the number of instructions is doubled (from 100k to 200k) and tripled (from 100k to 300k), while the number of clusters needed to cover an increasing percentage of the data is only going up 1.8 times (when the number of instructions doubles) and 2.6 times (when the number of instructions triples). Those figures reflect the effectiveness of the deduplication method for there is proportionately less to do when increasing the corpus size.

From the table we could see that 32% to 41% instructions (depending on the subset size) are detected as near-duplicates by only applying LSH. If we keep one instruction per cluster and combine out-of-cluster instructions together to form a new dataset, the size of the new one is reduced about 30-40% and the new dataset is assumed not to contain high lexical overlap instructions. This saves quite a lot of work for annotation since the annotator only have to label the dataset with more than 30% data removed. When we turn to the with-LSA model, the detected instructions nearly doubled compared to only-LSH model. However, one thing should be understood that the additional detected instructions are mostly assumed to be semantically similar. Unlike the result of LSH, those high lexical overlap instructions are almost certain to be near-duplicates while the result of the LSA is not. By bringing in the semantic information, instructions with variations and with similar structures or patterns are detected such as the near-duplicate pair example we made before *Form the pastry into a ball* and *Shape a dough into a plait*. Though being able to detect those desired near-duplicates, some noises are included into clusters. As shown in the type/token ratio for the with-LSA model, the ratio is really low compared to the only-LSH model which indicates that instructions with different variations are included. Some of these various instructions are not correctly related to the original cluster. Due to the time limit, we do not conduct experiments to deeply investigate the noise instructions detected by the with-LSA model. To use the with-LSA model for deduplication, one should be careful with noises in them.

Chapter 5

Conclusion

5.1 Summary

This project has investigated two different compositional models' ability of semantic composition on the task of aspectual classification. Two classification distinctions are explored, the first one is between event and state while the second one being between telic and atelic. Unlike many previous work on this classification problem, our model does not exploit manually constructed linguistic features, it only takes advantage of distributional vectors. As has been reported, the simple addition model outperforms the PLF model in every experiment. Though being less effective, the PLF model produces competitive results with limited training data. When making decision on the event/state classification, the *aux* seems to be the most contributing part apart from the main verb. Our best performed model achieves a state-of-the-art on this classification task. We have extended the original telicity dataset to a more balanced dataset with verbs being mostly ambiguous by adding self-annotated atelic sentences. We have shown that both *nsubj* and *dobj* have impacts on deciding the atelic/telic classification. After evaluating the deduplication process on subsets with various sizes of the MIT recipe dataset, the method is proved to be effective on detecting near-duplicates that have high lexical overlap or are semantically similar. Though being able to detect desired semantically similar near-duplicates, the with-LSA model brings in some noise instructions which should be carefully handled. The result of this project reflects the ability of capturing semantic interactions among different sentence components for compositional distributional semantics models. It also proves the effectiveness of learned distributional vectors for their ability to represent both syntactic and semantic meanings.

5.2 Current limitations

Due to the limited time and computing resources, our PLF model is trained on the SitEnt and telicity dataset rather than on the suggested corpora (the concatenation of ukWaC, Wikipedia and British National Corpus). Therefore, phrase vectors and functional matrices are poorly estimated on the small sized dataset for their low occurrences. This might be the main reason why the PLF model could not outperform the simple addition model.

Many preliminary work utilize stanford dependency parser and POS tagger like the relation extraction and the conjunction replacement. However, the tagger is not very accurate. Since the accuracy of the parser also depends on the tagging result, the parser also makes errors from time to time. When processing imperative sentences from MIT recipe dataset, the tagger often gives wrong tag to verbs which leads to wrong NP conjunction replacement and action conjunction split.

There are some other experiments that we do not have time explore. The additional near-duplicates given by with-LSA model contain some noises, we have not conducted experiments to find out why being detected. We only run limited subset sizes to evaluate the stability of the deduplication process which should be tested for larger subsets.

5.3 Future work

There are a number of things that would be explored if had time in future:

- PLF parameter training: In future, the phrase vectors and functional matrices for specific relations will be re-estimated on a much larger concatenated corpus (ukWaC, Wikipedia and British National Corpus) to properly investigate the performance of the PLF model.

- POS tagger correcting: As the POS tagger plays an important role in our experiments, it is necessary to correct the wrongly tagged results. In future, two possible correcting strategies will be explored. The first one is to correct on the produced results. We could try to define complex rules to automatically correct the wrongly tagged verb. The second one is to correct during training or decision making process. We could try to make the POS tagger to query a verb model before making the decision. Imperative sentences in the training data for the current Stanford tagger are less than 400 hundreds. We could also add more

imperative sentences to the training data.

- LSA noises tackling: Currently, we apply TF-IDF transformation on the term-document matrix before applying LSA. Since most instructions from the recipe dataset are short, TF-IDF representation might not make sense. Noises might be brought in by the problematic transformation. In future, we would like to investigate this problem by replacing TF-IDF transformation with other techniques. The LSA is also another exploration direction. We might try to explore several more common topic modelling methods like LDA to seek any reduction of noises.

- nsubj patterns: We find that the prosg/propl replacement strategy improves the importance of noun subjects compared to the proall replacement strategy. Apart from the assumption of the contribution made by the number of subjects, we would like to select sentences with specific subject pattern and conduct some comparison experiments to explore potential impacts.

- Annotation and aspectual classification on MIT recipe corpus: After conducting experiments on the SitEnt and telicity dataset, the next step would be investigating whether those patterns found on above experiments match potential patterns in the large single-domain MIT instruction corpus. Firstly, the annotation of aspectual property on the instruction corpus should be done. Then, we will apply the same two compositional models on it to see if there is any consistency.

Bibliography

Hakan Ancin, Rajashekhar Goli, Ankita Bakshi, Kumar Maddali, Joy Thomas, and Karthik Ramachandran. Composite locality sensitive hash based processing of documents, August 14 2012. US Patent 8,244,767.

Maria Antoniak and David Mimno. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics*, 6:107–119, 2018.

Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics, 2010.

William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 546–556. Association for Computational Linguistics, 2012.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

Maja Buljan, Sebastian Padó, and Jan Šnajder. Lexical substitution for evaluating compositional distributional models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 206–211, 2018.

Freddy YY Choi, Peter Wiemer-Hastings, and Johanna Moore. Latent semantic analysis for text segmentation. In *Proceedings of the 2001 conference on empirical methods in natural language processing*, 2001.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. A supervised approach to extractive summarisation of scientific papers. *arXiv preprint arXiv:1706.03946*, 2017.

Francisco Costa and António Branco. Aspectual type and temporal relation classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 266–275. Association for Computational Linguistics, 2012.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

Georgiana Dinu, Marco Baroni, et al. Dissect-distributional semantics composition toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, 2013.

David R Dowty. *Word Meaning and Montague Grammar: The Semantics of Verbs and Times in Generative Semantics and in Montague's PTO*. Reidel, 1979.

Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber. Universal dependencies to logical forms with negation scope. *arXiv preprint arXiv:1702.03305*, 2017.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. Learning distributions over logical forms for referring expression generation. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1914–1925, 2013.

Annemarie Friedrich and Damyana Gateva. Classification of telicity using cross-linguistic annotation projection. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2559–2565, 2017.

Annemarie Friedrich and Alexis Palmer. Automatic prediction of aspectual class of verbs in context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 517–523, 2014.

Annemarie Friedrich and Manfred Pinkal. Automatic recognition of habituals: a three-way classification of clausal aspect. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2471–2481, 2015.

Annemarie Friedrich, Alexis Palmer, and Manfred Pinkal. Situation entity types: automatic classification of clause-level aspect. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1757–1768, 2016.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. Multi-step regression learning for compositional distributional semantics. *arXiv preprint arXiv:1301.6939*, 2013.

Abhijeet Gupta, Jason Utt, and Sebastian Padó. Dissecting the practical lexical function model for compositional distributional semantics. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 153–158, 2015.

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the ACL 2010 conference short papers*, pages 68–73. Association for Computational Linguistics, 2010.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

Thomas Kober, Nathanael Chambers, and Mark Steedman. Modelling aspectual class of verbs with distributional semantics. 2018.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

Alex Lascarides and Nicholas Asher. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and philosophy*, 16(5):437–493, 1993.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279. Association for Computational Linguistics, 2004.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244, 2008.

Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational linguistics*, 14(2):15–28, 1988.

Dan I Moldovan and Vasile Rus. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 402–409. Association for Computational Linguistics, 2001.

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith. A sequencing model for situation entity classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 896–903, 2007.

Denis Paperno, Marco Baroni, et al. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 90–99, 2014.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Christopher Quirk. Training a sentence-level machine translation confidence measure. In *LREC*. Citeseer, 2004.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Samuel Ritter, Cotie Long, Denis Paperno, Marco Baroni, Matthew Botvinick, and Adele Goldberg. Leveraging preposition ambiguity to assess compositional distributional models of semantics. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 199–204, 2015.

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Eric V Siegel and Kathleen R McKeown. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics*, 26(4):595–628, 2000.

Carlota S Smith. The parameter of aspect, vol. 43 of studies in linguistics and philosophy, 1991.

Carlota S Smith. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press, 2003.

Carlota S Smith. Aspectual entities and tense in discourse. In *Aspectual inquiries*, pages 223–237. Springer, 2005.

Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003.

Zeno Vendler. Verbs and times. chapter 4 in linguistics in philosophy, pp97-121, 1967.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. Event representations with tensor-based compositions. *arXiv preprint arXiv:1711.07611*, 2017.

Jianming Zheng, Yupu Guo, Chong Feng, and Honghui Chen. A hierarchical neural-network-based document representation approach for text classification. *Mathematical Problems in Engineering*, 2018, 2018.