Tense Sense Disambiguation: a New Syntactic Polysemy Task

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Polysemy is a major characteristic of natural languages. Like words, syntactic forms can have several meanings. Understanding the correct meaning of a syntactic form is of great importance to many NLP applications. In this paper we address an important type of syntactic polysemy – the multiple possible senses of tense syntactic forms. We make our discussion concrete by introducing the task of *Tense Sense Disambiguation* (TSD): given a concrete tense syntactic form present in a sentence, select its appropriate sense among a set of possible senses. Using English grammar textbooks, we compiled a syntactic sense dictionary comprising common tense syntactic forms and semantic senses for each. We annotated thousands of BNC sentences using the defined senses. We describe a supervised TSD algorithm trained on these annotations, which outperforms a strong baseline for the task.

1 Introduction

The function of syntax is to combine words to express meanings, using syntactic devices such as word order, auxiliary words, and morphology (Goldberg, 1995). Virtually all natural language devices used for expressing meanings (e.g., words) exhibit polysemy. Like words, concrete syntactic forms (the sentence words generated by specific syntactic devices) can have several meanings. Consider the following sentences:

(a) They **are playing** chess in the park.

(b) They **are playing** chess next Tuesday.

Both contain the concrete syntactic form 'are playing', generated by the abstract syntactic form usually known as 'present progressive' (am/is/are + V-ing). In (a), the meaning is 'something happening now', while in (b) it is 'a plan to do something in the future'. Note that the polysemy is of the syntactic form as a unit, not of individual words. In particular, the verb 'play' is used in the same sense in both cases.

In this paper we address a prominent type of syntactic form polysemy: the multiple possible senses that tense syntactic forms can have. Disambiguating the polysemy of tense forms is of theoretical and practical importance (Section 2). To make our discussion concrete, we introduce the task of *Tense Sense Disambiguation* (TSD): given a concrete tense syntactic form in a sentence, select its correct sense among a given set of possible senses (Section 3).

The disambiguation of polysemy is a fundamental problem in NLP. For example, Word Sense Disambiguation (WSD) continues to attract a large number of researchers (Agirre and Edmonds, 2006). TSD has the same structure as WSD, with different disambiguated entities.

For experimenting with the TSD task, we compiled an English syntactic sense dictionary based on a thorough study of three major English grammar projects (Section 4). We selected 3000 sentences from the British National Corpus containing 4702 concrete syntactic forms, and annotated each of these by its sense (Section 5).We developed a supervised learning TSD algorithm that uses various feature types and takes advantage of the task structure (Section 6). Our algorithm substantially outperforms the ‘most frequent sense’ baseline (Section 7).

TSD is fundamental to sentence understanding and thus to NLP applications such as textual inference, question answering and information retrieval. To the best of our knowledge, this is the first paper to address this task. In Section 8 we discuss research directions relevant to TSD placing the new task in the context of the previous research of syntactic ambiguity resolution.

2 TSD Motivation

In this work we follow linguistics theories that posit that tense does not directly reflect conceptual time as one might think. Dinsmore (1991) and Cutrer (1994) explain that the same tense may end up indicating very different objective time relations relative to the sentence production time.

Fauconnier (2007) exemplifies such phenomena. In the following sentences, the present tense corresponds to the *future* time: (1) The boat leaves next week. (2) When he comes tomorrow, I will tell him about the party. (3) If I see him next week, I will ask him to call you.

In contrast, the following present tense sentences talk about events that happened in the *past*: (1) I am walking down the street one day when suddenly this guy walks up to me. (2) He catches the ball. He runs. He makes a touchdown. (morning-after sports report).

Another set of examples is related to the past tense. In the following sentences it corresponds to a *present* time: (1) Do you have a minute? I wanted to ask you a question. (2) I wish I lived closer to my family now. In contrast, in the following two sentences, it corresponds to a future time: (1) If I had the time next week, I would go to your party. (2) I cannot go to the concert tonight. You will have to tell me how it was.

Fauconnier explains these phenomena by a model for the grammar of tense. According to this model, the grammar specifies partial constraints on time and fact/prediction status that hold locally between mental spaces within a discourse configuration. We may obtain actual information about time by combining this with other available pragmatic information. Accordingly, the same tense may end up indicating very different objective time relations relative to the speech event.

TSD fits well with modern linguistics theories. For example, in the construction grammar framework (Goldberg, 1995), the ‘construction’ is the basic unit, comprised of a form and a meaning. Words, multiword expressions, and syntactic forms are all valid constructions. It is thus very natural to address the sense disambiguation problem for all of these. In this paper we focus on tense constructions.

For many NLP applications, it is very important to disambiguate the tense forms of the sentence. Among these applications are: (1) machine translation, as the actual time described by one tense form in the source language may be described by a different tense form in the target language; (2) understanding the order of events in a text; (3) textual entailment, when the optional entailed sentences refer to the time and/or order of events of the source sentence. Many more examples also exist.

3 The TSD Task

In this section we formally define the TSD task, discuss its nature vs. WSD, and describe various concrete task variants.

Task definition. First, some essential terminology. The function of syntax is to combine lexical items (words, multiword expressions) to express meanings. This function is achieved through syntactic *devices*. The most common devices in English are word order, morphology, and the usage of auxiliary words. An *Abstract Syntactic Form (ASF)* is a particular set of devices that can be used to express a set of meanings. A *Concrete Syntactic Form (CSF)* is a concrete set of words generated by an ASF for expressing a certain meaning in an utterance[1]. A CSF is *ambiguous* if its generating ASF has more than one meaning, which is the usual case. In this case we also say that the ASF is ambiguous.

Here are a few examples. The ‘present progressive’ ASF has the form ‘am/is/are V-ing’[2], which employs all three main devices. It is ambiguous,

[1]In some linguistic theories, the central notion is the *construction*, which combines an ASF (referred to as the form of the construction) with a single meaning (Goldberg, 1995).

[2]Note that strictly speaking, these are three different ASFs. We refer to this ASF family by a single name because they have the same set of meanings and because it is standard to treat them as a single ASF.

as shown in Section 1. The 'present simple' ASF has the form 'V(+s)'[3], and is ambiguous as well: in the sentence 'My Brother arrives this evening', the CSF 'arrives' conveys the meaning of 'a future event arranged for a definite time', while in the sentence 'The sun rises in the East' the meaning is that of a repeated event.

TSD vs. WSD. The TSD task is to disambiguate the semantic sense of a tense syntactic form. TSD is clearly different from WSD. This is obvious when the CSF comprises two words that are not a multi-word expression, and is usually also the case when it comprises a single word. Consider the 'My Brother arrives this evening' example above. While the verb 'arrive' has two main senses: 'reach a place', and 'begin', as in 'Summer has arrived', in that example we focused on the disambiguation of the tense sense of the 'arrives' construction.

Concrete task variants. Unlike with words, the presence of a particular CSF in a sentence is not trivially recognizable. Consequently, there are three versions of the TSD task: (1) we are given the sentence, a marked subset of its words comprising a CSF, and the ASF that has generated these words; (2) we are given the sentence and a marked subset of its words comprising a CSF, without knowing the generating ASF; (3) we are given only the sentence and we need to find the contained CSFs and their ASFs. In all cases, we need to disambiguate the sense of the ASFs. We feel that the natural granularity of the task is captured by version (2). However, since the ASF can usually be identified using relatively simple features, we also report results for version (1). The main difficulty in all versions is identifying the appropriate sense, as is the case with WSD.

4 The Syntactic Sense Dictionary

A prerequisite to any concrete experimentation with the TSD task is a syntactic sense dictionary. Based on a thorough examination of three major English grammar projects, we compiled a set of 18 common English tense ASFs and their possible senses. The projects are (1) the Cambridge University Press English Grammar In Use series, comprising three books (essential, intermediate and advanced) (Murphy, 2007; Murphy, 1994; Hewings, 2005); (2) the English grammar texts resulting from the seminal corpus-based Cobuild project (elementary, advanced) (Willis and Wright, 2003; Willis, 2004); (3) the Longman Grammar of Spoken and Written English (Biber et al., 1999).

As in any sense dictionary, in many cases it is hard to draw the line between senses. In order to be able to explore the computational limits of the task, we have adopted a policy of fine sense granularity. For example, senses 1 and 3 of the 'present simple' ASF in Table 1 can be argued to be quite similar to each other, having a very fine semantic distinction. A specific application may choose to collapse some senses into one.

We used the conventional ASF names, which should not be confused with their meanings (e.g., the 'present simple' ASF can be used to refer to *future*, not present, events, as in Table 1, sense 4).

The ASF set thus obtained is: real conditionals, hypothetical conditionals, wishes, reported speech, present simple, present progressive, present perfect, present perfect progressive, past simple, past progressive, past perfect, past perfect progressive, 'be + going + to + infinitive', future progressive, future perfect, future perfect progressive, 'would' tense forms, and 'be + to + infinitive'. Note that the first four ASFs are not direct tense forms; we include them because they involve tensed sub-sentences whose disambiguation is necessary for disambiguation of the whole ASF. The total number of possible senses for these 18 ASFs is 103.

Table 1 shows the complete senses set for the 'present simple' and 'be + to + infinitive' ASFs, plus an example sentence for each sense. Space limitations prevent us from listing all form senses here; we will make the listing available online.

5 Corpus Creation and Annotation

We selected 3000 sentences from the British National Corpus (BNC) (Burnard, 2000), containing 4702 CSFs (1.56 per sentence). These sentences with their CSFs were sense annotated. To select the 3000 sentences, we randomly sampled sentences from the various written and spoken sections of the

[3]Again, these are two ASFs, one adding an 's' and one using the verb as is.

	Present Simple
1	*Things that are always true* It gets cold in the winter.
2	*Regular and repeated actions and habits* My parents often eat meat.
3	*General facts* Mr. Brown is a teacher.
4	*A future event arranged for a definite time* The next train arrives at 11:30.
5	*Plans, expectations and hopes* We hope to see you soon.
6	*Ordering someone to do something* Take your hands out of your pockets!
7	*Something happening now, with verbs that are not used in the present progressive in this sense* I do not deny the allegation.
8	*Events happening now (informal; common in books, scripts, radio etc.)* She goes up to this man and looks into his eyes.
9	*Past actions* I was sitting in the park reading a newspaper when all of a sudden this dog jumps at me.
10	*Newspaper headlines, for recent events* Quake hits central Iran.
11	*When describing the content of a book* Thompson gives an exhaustive list in chapter six.
	‘be + to + infinitive’
1	*Events that are likely to happen in the near future* Police officers are to visit every home in the area.
2	*Official arrangements, formal instructions & orders* You are not to leave without my permission.
3	*In an if-clause to say that something must happen before something else can happen* If the human race is to survive, we must look at environmental problems now.

Table 1: The full set of senses of the ‘present simple’ and ‘be + to + infinitive’ abstract syntactic forms (ASFs), with an example for each.

corpus, giving each section an equal weight. To guarantee ample representation of ASFs, we manually defined auxiliary words typical of each ASF (e.g., ‘does’, ‘been’ etc), and sampled hundreds of sentences for each set of these auxiliary words. To make sure that our definition of auxiliary words does not skew the sampling process, and to obtain ASFs that do not have clear auxiliary words, we have also added 1000 random sentences. The number of CSF instances obtained for each ASF ranges from 100 (future perfect) to over 850 (present simple). All senses are represented; the number of senses represented by at least 15 CSFs is 77 (out of 103, average number of CSFs per sense is 45.65).

We implemented an interactive application that displays a sentence and asks an annotator to (1) mark words that participate in the CSFs contained in the sentence; (2) specify the ASF(s) of these CSFs; and (3) select the appropriate ASF sense from the set of possible senses. Annotators could also indicate ‘none of these senses’, which they did for 2.6% (122 out of 4702) of the CSFs.

Annotation was done by two annotators (university students). To evaluate inter-annotator agreement, a set of 210 sentences (7% of the corpus), containing at least 10 examples of each ASF, was tagged by both annotators. The CSF+ASF identification inter-annotator agreement was 98.7%, and the inter-annotator agreement for the senses was 84.2%. We will make the annotated corpus and annotation guidelines available online.

6 Learning Algorithm

In this section we describe our learning model for the TSD task. First, note that the syntactic sense is not easy to deduce from readily computable annotations such as the sentence’s POS tagging, dependency structure, or parse tree (see Section 8). Hence, a learning algorithm is definitely needed.

As common in supervised learning, we encode the CSFs into feature vectors and then apply a learning algorithm to induce a classifier. We first discuss the feature set and then the algorithm.

Features. We utilize three sets of features: basic features, lexical features, and a set of features based on part-of-speech (POS) tags (Table 2). The ‘auxiliary words’ referred to in the table are the manually specified words for each ASF that have assisted us in sampling the corpus (see Section 5). ‘Content words’ are the non-auxiliary words appearing in the CSF[4]. Content words are usually verbs, since we focus here on tense-related ASFs. The position and distance of a form are based on its leftmost word (auxiliary or content).

The personal pronouns used in the position features are: I, you, he, she, it, they, and we. For

[4]Usually, there is a single content word. However, there may be more than one, e.g. for phrasal verbs.

simplicity, we considered every word starting with a capital letter that is not the first word in the sentence to be a name.

Each 'Conditional' CSF contains two tense CSFs. The one that is not the CSF currently encoded by the features is referred to as its 'mate'.

For the time lexical features we used 16 words (e.g., recently, often, now). For the reported speech lexical features we used 14 words (e.g., said, replied, wrote[5]). The words were obtained from the grammar texts and our corpus development set.

The POS tagset used by the POS-based features is that of the WSJ PennTreebank (see Section 7). The possible verb tags in this tagset are: VB for the base form, VBD for past tense, VBN for past participle, VBG for a present participle or gerund (-ing), VBP for present tense that is not 3rd person singular, and VBZ for present simple 3rd person singular.

Conjunctions and prepositions are addressed through the POS tags CC and IN. Using the PRP tag to detect pronouns or lexical lists for conjunctions and prepositions yielded no significant change in the results.

In Section 7 we explore the impact each of the feature sets has on the performance of the algorithm. Our results indicate that the basic features have the strongest impact, the POS-based features enhance the performance in specific cases and the lexical features only marginally affect the final results.

Algorithm. Denote by x_i the feature vector of a CSF instance i, by C_i the set of possible labels for x_i, and by $c_i \in C_i$ the correct label. The training set is $\{(x_j, C_j, c_j)\}_{j=1}^{n}$. Let (x_{n+1}, C_{n+1}) be a test CSF. As noted in Section 3, there are two versions of the task, one in which C_i includes the totality of sense labels, and one in which it includes only the labels associated with a particular ASF. In both cases, the task is to select which of the labels in C_{n+1} is its correct label c_{n+1}.

Owing to the task structure, it is preferable to use an algorithm that allows us to restrict the possible labels of each CSF. For both task versions, this would help in computing better probabilities during the training stage, since we know the ASF type of training CSFs. For the task version in which the ASF type is known at test time, this would also help during the test stage.

Basic Features
Form words. Auxiliary and content words of the CSF.
Form type. The type, if it is known during test time.
Other forms. The auxiliary and content words (and type, if known) of the other CSFs present in the sentence.
Position. The position of the CSF in the sentence, its distance from the end of the sentence, whether it is in the first (last) three words in the sentence, its distance from the closest personal pronoun or name.
Wish. Is there a CSF of type 'wish' before the encoded form, the number of CSFs between that 'wish' form and the encoded CSF (if there are several such 'wish' forms, we take the closest one to the encoded form).
Conditional. Does the word 'if' appear before the encoded form, is the 'if' the first word in the sentence, the number of CSFs between the 'if' and the encoded form, the auxiliary and content words (and type, if known) of the mate form, is there a comma between the encoded form and its mate form, does the word 'then' appear between the encoded form and its mate form.
Punctuation. The type of end of sentence marker, distance of the encoded form from the closest predecessor (successor) comma.

Lexical Features
Time. Time words appearing in the sentence, if any.
Reported speech. Reported speech words appearing in the sentence, if any.
Be. Does the encoded form contain the verb 'be'.

Features Based on POS Tags
*Form.*The POS of the verb in the encoded form.
Other forms. The POS of the verb in the other CSFs in the sentence.
POS tags. The POS tags of the two words to the left (right) of the encoded form.
Conjunction POS. Is there a Conjunction (CC) between the encoded form and its closest predecessor (successor) form, the distance from that conjunction.
Preposition POS. Is there a Preposition (IN) between the encoded form and its closest predecessor (successor) form, the distance from that preposition.

Table 2: Basic features (top), lexical features (middle) and POS tags-based features (bottom) used by the TSD classifier.

For the version in which ASF type is known at test time, we experimented in two scenarios. In the first,

[5]These are all in a past form due to the semantics of the reported speech form.

we take the ASF type at test time from the manual annotation and provide it to the algorithm. In the second, instead of the manual annotation, we implemented a simple rule-based classifier for selecting ASF types. The classifier decides what is the type of an ASF according to the POS tag of its verb and to its auxiliary words (given in the annotation). For example, if we see the auxiliary phrase 'had been' and the verb POS is not VBG, then the ASF is 'past perfect simple'. This classifier's accuracy on our development (test) data is 94.1 (91.6)%. In this scenario, when given a test CSF, X_{n+1}, its set of possible labels C_{n+1} is defined by the classifier output. In the features in which ASF type is used (see table 2), it is taken from the classifier output in this case.

The sequential model algorithm presented by Even-Zohar and Roth (2001) directly supports this label restriction requirement [6]. We use the SNOW learning architecture for multi-class classification (Roth, 1998), which contains an implementation of that algorithm. The SNOW system allows us not to define restrictions if so desired. It also lets us choose the learning algorithm used when it builds its classifier network. The algorithm can be Perceptron (MacKay, 2002), Winnow (Littlestone, 1988) or Naive Bayes (MacKay, 2002)[7]. In Section 7 we analyze the effect that these decisions have on our results.

Classifier Selection. Investigating the best configuration of the SNOW system with development data, we found that Naive Bayes gave the best or close to best result in all experimental conditions. We therefore report our results when this algorithm is used. Naive Bayes is particularly useful when relatively small amounts of training CSF instances are available (Zhang, 2004), and achieves good results when compared to other classifiers for the WSD task (Mooney, 1996), which might explain our results. Fine tuning of Winnow parameters also leads to high performance (sometimes the best), but most other parameter configurations lead to disappointing results. For the Perceptron, most parameter configurations lead to good results (much better than the baseline), but these were a few percent worse than the best Winnow or Naive Bayes results.

[6]Note that the name of the learning algorithm is derived from the fact that it utilizes classifiers to sequentially restrict the number of competing classes while maintaining with high probability the presence of the true outcome. The classification task it performs is not sequential in nature.

[7]Or a combination of these algorithms, which we did not explore in this paper.

7 Experimental Results

Experimental setup. We divided the 3000 annotated sentences (containing 4702 CSFs) to three datasets: training data (2100 sentences, 3183 forms), development data (300 sentences, 498 forms) and test data (600 sentences, 1021 forms). We used the development data to design the features for our learning model and to tune the parameters of the SNOW sequential model. In addition we used this data to design the rules of the ASF type classifier (which is not statistical and does not have a training phase).

For the POS features, we induced POS tags using the MXPOST POS tagger (Ratnaparkhi, 1996). The tagger was trained on sections 2-21 of the WSJ PennTreebank (Marcus et al., 1993) annotated with gold standard POS tags. We used a publicly available implementation of the sequential SNOW model[8].

We experimented in three conditions. In the first (TypeUnknown), the ASF type is not known at test time. In the last two, it is known at test time. These two conditions differ in whether the type is taken from the gold standard annotation of the test sentences (TypeKnown), or from the output of the simple rule-based classifier (TypeClassifier, see Section 6). For both conditions, the results reported below are when both ASF type features and possible labels sets are provided during training by the manual annotation. This is true also for the training of the MFS baseline (see below)[9].

We report an algorithm's quality using accuracy, that is, the number of test CSFs that were correctly resolved by the algorithm divided by the total number of test CSFs.

Baseline. We compared the performance of our algorithm to the 'most frequent sense' (MFS) base-

[8]http://l2r.cs.uiuc.edu/∼cogcomp/asoftware.php?skey=SNOW

[9]For the TypeClassifier condition, we also experimented using an ML technique that sometimes reduces noise, where training is done using the classifier types. We obtained very similar results to those reported.

	TypeUnknown	TypeClassifier	TypeKnown
Our algorithm	**49.7%**	**58.8%**	**62%**
MFS baseline	13.5%	42.9%	46.7%

Table 3: Performance of our algorithm and of the MFS baseline where at test time ASF type is known (right), unknown (left) or given by a simple rule-based classifier (middle). Our algorithm is superior in all three conditions.

	Constrained Model		Unconstrained Classifier	
	All features	Base+Lexical features	All features	Base+Lexical features
Type features	57.9%	57.7%	53%	50.1%
No type features	57.2%	55.4%	48%	42.6%

Table 4: Impact of POS features. When the constrained model is used (left section), POS features have no effect on the results when ASF type information is encoded. When an unconstrained classifier is used, POS features affect the results both when ASF type features are used and when they are not (see discussion in the text).

line. This baseline is common in semantic disambiguation tasks and is known to be quite strong. In the condition where the ASF type is not known at test time, MFS gives each form in the test set the sense that was the overall most frequent in the training set. That is, in this case the baseline gives all test set CSFs the same sense. When the ASF type is known at test time, MFS gives each test CSF the most frequent sense *of that ASF type* in the training set. That is, in this case all CSFs having the same ASF type get the same sense, and forms of different types are guaranteed to get different senses.

Recall that the condition where ASF type is known at test time is further divided to two conditions. In the TypeKnown condition, MFS selects the most frequent sense of the manually created ASF type, while in the TypeClassifier condition it selects the most frequent sense of the type decided by the rule-based classifier. In this condition, if the classifier makes a mistake, MFS will necessarily make a mistake as well.

Note that a random baseline which selects a sense for every test CSF from a uniform distribution over the possible senses (103 in our case) would score very poorly.

Results. Table 3 shows our results. Results are shown where ASF type is not known at test time (left), when it is decided at test time by a rule-based classifier (middle) and when it is known at test time (right). Our algorithm outperforms the MFS baseline in all three conditions. As expected, both our algorithm and the MFS baseline perform better when ASF type information is available at test time (TypeClassifier and TypeKnown conditions), and improve as this data becomes more accurate (the TypeKnown condition)[10].

Analyzing the per-type performance of our algorithm reveals that it outperforms the MFS baseline for each and every ASF type. For example, in the TypeKnown condition, the accuracy gain of our algorithm over the baseline[11] varies from 4% for the ‘present perfect’ to 30.6% and 29.1% for the ‘past perfect’ and ‘present simple’ ASFs.

Below we analyze the roles of the different components of our learning algorithm in performing the TSD task. Since this is the first exploration of the task, it is important to understand what properties are essential for achieving good performance. The analysis is done by experimenting with development data, and focuses on the TypeKnown and TypeUnknown conditions. Patterns for the TypeClassifier condition are very similar to the patterns for the TypeKnown condition.

The Possible Senses Constraint. We use the learning model of Even-Zohar and Roth (2001), which allows us to constrain the possible senses an input vector can get to the senses of its ASF type. We ran our model without this constraint during both training and test time (recall that for the above results, this constraint was always active during training). In this case, the only difference between the TypeKnown and the TypeUnknown conditions is whether ASF type features are encoded at test time. In the TypeKnown condition, the accuracy of the algorithm drops from 57.9% (when using training and test time constraints and ASF type features) to 53% (when using only ASF type features but no constraints). In the TypeUnknown condition, accuracy drops from 57.24% (when using training time constraints) to 48.03% (when neither constraints nor ASF type features are used). Note

[10]Recall that the performance of the rule-based ASF type classifier on test data is not 100% but 91.6% (Section 6).

[11]$accuracy(algorithm) - accuracy(MFS)$.

that the difference between the constrained model and the unconstrained model is quite large.

The MFS baseline achieves on development data 42.9% and 13.2% in the TypeKnown and TypeUnknown conditions respectively[12]. Thus, the algorithm outperforms the baseline both when the constrained model is used and when an unconstrained multi-class classifier is used.

Note also that when constraints on the possible labels are available at training time, test time constraints and ASF type features (whose inclusion is the difference between the TypeKnown and TypeUnknown) have a minor effect on the results (57.9% for TypeKnown compared to 57.24% for TypeUnknown). However, when training time constraints on the possible labels are not available at training time, ASF type features alone do have a significant effect on the result (53% for TypeKnown compared to 48.03% for TypeUnknown).

POS Features. We next explore the impact of the POS features on the results. These features encode the inflection of the verbs in the CSF, as well as the POS tags of the two words to the left and right of the CSF.

Verb forms provide some partial information corresponding to the ASF type features encoded at the TypeKnown scenario. Table 4 shows that when both label constraints and ASF type features are used, POS features have almost no impact on the final results. When the constrained model is used but ASF type features are not encoded, POS features have an effect on the results. We conclude that when using the constrained model, POS features are important mainly for ASF type information. When the unconstrained classifier is used, POS features have an effect on performance whether ASF type features are encoded or not. In the last case the impact of POS features is larger. In other words, when using an unconstrained classifier, POS features give more than ASF type information to to the model.

Lexical Features. To explore the impact of the lexical features, we removed the following features: time words, reported speech words and 'be' indication features. We saw no impact on model performance when using the constrained model, and a 0.5% decrease when using the unconstrained classifier. That is, our model does not require these lexical features, which is somewhat counter-intuitive. Lexical statistics may turn out to be helpful when using a much larger training set.

Conditional and Wish Features. The conditionals and 'wish' features have a more substantial impact on the results, as they have a role in defining the overall syntactic structure of the sentence. Discarding these features leads to 4% and 1.4% degradation in model accuracy when using the constrained and unconstrained models respectively.

8 Relevant Previous Work

As far as we know, this is the first paper to address the TSD task. In this section we describe related research directions and compare them with TSD.

A relevant task to TSD is WSD (Section 1 and Section 3). Many algorithmic approaches and techniques have been applied to supervised WSD (for reviews see (Agirre and Edmonds, 2006; Mihalcea and Pedersen, 2005; Navigli, 2009)). Among these are various classifiers, ensemble methods combining several supervised classifiers, bootstrapping and semi-supervised learning methods, using the Web as a corpus and knowledge-based methods relying mainly on machine readable dictionaries. Specifically related to this paper are works that exploit syntax (Martinez et al., 2002; Tanaka et al., 2007) and ensemble methods (e.g. (Brody et al., 2006)) to WSD. The references above also describe some unsupervised word sense induction algorithms.

Our TSD algorithm uses the SNOW algorithm, which is a sparse network of classifiers (Section 6). Thus, it most resembles the ensemble approach to WSD. That approach has achieved very good results in several WSD shared tasks (Pedersen, 2000; Florian and Yarowsky, 2002).

Since temporal reasoning is a direct application of TSD, research on this direction is relevant. Such research goes back to (Passonneau, 1988), which introduced the PUNDIT temporal reasoning system. For each tensed clause, PUNDIT first decides whether it refers to an actual time (as in 'We flew TWA to Boston') or not (as in 'Tourists flew TWA to Boston', or 'John always flew his own plane to Boston'). The temporal structure of actual time

[12]Note that these numbers are for development data only.

clauses is then further analyzed. PUNDIT's classification is much simpler than in the TSD task, addressing only actual vs. non-actual time. PUNDIT's algorithmic approach is that of a Prolog rule based system, compared to our statistical learning corpus-based approach. We are not aware of further research that followed their sense disambiguation direction.

Current temporal reasoning research focuses on temporal ordering of events (e.g., (Lapata, 2006; Chambers and Jurafsky, 2008)), for which an accepted atomic task is the identification of the temporal relation between two expressions (see e.g., the TempEval task in SemEval '07 (Verhagen et al., 2007)). This direction is very different from TSD, which deals with the semantics of *individual* concrete tense syntactic forms. In this sense, TSD is an even more atomic task for temporal reasoning.

A potential application of TSD is machine translation where it can assist in translating tense and aspect. Indeed several papers have explored tense and aspect in the MT context. Dorr (1992) explored the integration of tense and aspect information with lexical semantics for machine translation. Schiehlen (2000) analyzed the effect tense understanding has on MT. Ye and Zhang (2005) explored tense tagging in a cross-lingual context. Ye et al., (2006) extracted features for tense translation between Chinese and English. Murata et al., (2007) compared the performance of several MT systems in translating tense and aspect and found that various ML techniques perform better on the task.

Another related field is 'deep' parsing, where a sentence is annotated with a structure containing information that might be relevant for semantic interpretation (e.g. (Hajic, 1998; Baldwin et al., 2007)). TSD senses, however, are not explicitly represented in these grammatical structures, and we are not aware of any work that utilized them to do something close to TSD. This is a good subject for future research.

9 Conclusion and Future Work

In this paper we introduced the Tense Sense Disambiguation (TSD) task, defined as selecting the correct sense of a concrete tense syntactic form in a sentence among the senses of abstract syntactic forms in a syntactic sense dictionary. Unlike in other semantic disambiguation tasks, the sense to be disambiguated is not lexical but of a *syntactic* structure. We prepared a syntactic sense dictionary, annotated a corpus by it, and developed a supervised classifier for sense disambiguation that outperformed a strong baseline.

An obvious direction for future work is to expand the annotated corpus and improve the algorithm by experimenting with additional features. For example, we saw that seeing the full paragraph containing a sentence helps human annotators decide on the appropriate sense which implies that using larger contexts may improve the algorithm.

TSD can be a very useful operation for various high-level applications, for example textual inference, question answering, and information retrieval, in the same way that textual entailment (Dagan et al., 2006) was designed to be. In fact, TSD can assist textual entailment as well, since the sense of a tense form may provide substantial information about the relations entailed from the sentence. Using TSD in such applications is a major direction for future work.

References

Eneko Agirre and Philip Edmonds (Eds). 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer Verlag.

Timothy Baldwin, Mark Dras, Julia Hockenmaier, Tracy Holloway King, and Gertjan van Noord. 2007. The Impact of Deep Linguistic Processing on Parsing Technology. IWPT '07.

Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conard, Edward Finegan. 1999. *Longman Grammar of Spoken and Written English*. Longman.

Samuel Brody, Roberto Navigli and Mirella Lapata. 2006. Ensemble Methods for Unsupervised WSD. ACL-COLING '06.

Lou Burnard. 2000. *The British National Corpus User Reference Guide*. Technical Report, Oxford University.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly Combining Implicit Constraints Improves Temporal Ordering. EMNLP '08.

Michelle Cutrer. 1994. Time and Tense in Narratives and in Everyday Language. *PhD dissertation*, University of California at San Diego.

Ido Dagan, Oren Glickman and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science* 2006, 3944:177-190.

John Dinsmore. 1991. *Partitioned representations*. Dordrecht, Netherlands: Kluwer.

Bonnie Dorr. 1992. A Two-Level Knowledge Representation for Machine Translation: Lexical Semantics and Tense/Aspect. In James Pustejovsky and Sabine Bergler, editors, Lexical Semantics and Knowledge Representation.

Yair Even-Zohar and Dan Roth. 2001. A Sequential Model for Multi-Class Classification. EMNLP '01.

Gilles Fauconnier. 2007. *Mental Spaces*. in Dirk Geeraerts and Hubert Cuyckens, editors, The Oxford Handbook of Cognitive Linguistics.

Radu Florian and David Yarowsky. 2002. Modeling Consensus: Classifier Combination for Word Sense Disambiguation. EMNLP '02.

Adele E. Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

Jan Hajic. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. *Issues of Valency and Meaning*, 106–132.

Martin Hewings. 2005. *Advanced Grammar in Use, Second Edition*. Cambridge University University.

Mirella Lapata and Alex Lascarides. 2006. Learning Sentence-internal Temporal Relations. *Journal of Artificial Intelligence Research*, 27:85–117.

Nick Littlestone. 1988. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 285–318.

David MacKay. 2002. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David Martinez, Eneko Agirre, Lluis Marquez. 2002. Syntactic Features for High Precision Word Sense Disambiguation. COLING '02.

Rada Mihalcea and Ted Pedersen. 2005. Advances in Word Sense Disambiguation. Tutorial in ACL '05.

Raymond J. Mooney. 1996. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. EMNLP '96.

Masaki Murata, Qing Ma, Kiyotaka Uchimoto, Toshiyuki Kanamaru and Hitoshi Isahara. 2007. Japanese-to-English translations of Tense, Aspect, and Modality Using Machine-Learning Methods and Comparison with Cachine-Translation Systems on Market. LREC '07.

Raymond Murphy. 1994. *English Grammar In Use, Second Edition*. Cambridge University Press.

Raymond Murphy. 2007. *Essential Grammar In Use, Third Edition*. Cambridge University Press.

Roberto Navigli. 2009. Word Sense Disambiguation: a Survey. *ACM Computing Surveys*, 41(2) 1–69.

Rebecca J. Passonneau. 1988. A Computational Model of Semantics of Tenses and Aspect. *Computational Linguistics*, 14(2):44–60.

Ted Pedersen. 2000. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. NAACL '00.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. EMNLP '06.

Dan Roth. 1998. Learning to Resolve Natural Language Ambiguities: A Unified Approach. AAAI '98.

Michael Schiehlen. 2000. Granularity Effects in Tense Translation. COLING '00.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. ACL '07.

Takaaki Tanaka, Francis Bond, Timothy Baldwin, Sanae Fujita and Chikara Hashimoto. 2007. Word Sense Disambiguation Incorporating Lexical and Structural Semantic Information. EMNLP-CoNLL '07.

Dave Willis and Jon Wright. 2003. *Collins Cobuild Elementary English Grammar, Second Edition*. HarperCollins Publishers.

Dave Willis. 2004. *Collins Cobuild Intermediate English Grammar, Second Edition*. HarperCollins Publishers.

Yang Ye, Victoria Li Fossum and Steven Abney. 2006. Latent Features in Automatic Tense Translation between Chinese and English. SIGHAN '06.

Yang Ye and Zhu Zhang. 2005. Tense Tagging for Verbs in Cross-Lingual Context: A Case Study. IJCNLP '05.

Harry Zhang. 2004. The Optimality of Naive Bayes. FLAIRS '04.