Hierarchical Recursive Tagset for Annotating Cooking Recipes

Sharath Reddy Gunamgari*
Samsung Research India
Bangalore, India - 560037
gsharathreddy77
@gmail.com

Sandipan Dandapat
CNGL, DCU
Dublin 9, Ireland
sdandapat
@computing.dcu.ie

Monojit Choudhury
Microsoft Research India
Bangalore, India - 560001
monojitc
@microsoft.com

Abstract

Several natural language annotation schemas have been proposed for different natural language understanding tasks. In this paper we present a hierarchical and recursive tagset for annotating natural language recipes. Our recipe annotation tagset is developed to capture both syntactic and semantic information in the text. First, we propose our hierarchical recursive tagset that captures cooking attributes and relationships among them. Furthermore, we develop different heuristics to automatically annotate natural language recipes using our proposed tagset. These heuristics use surface-level and syntactic information from the text and the association between words. We are able to annotate the recipe text with 91% accuracy in an ideal situation.

1 Introduction

Cooking or cookery is the art or practice of preparing food for consumption. Our motivation is to aid the applications useful for cooking process like recipe search, recipe recommendation etc. The first step required for these applications is to extract the information from a given recipe by annotating that recipe. Annotating natural language text is an important step towards the process of natural language understanding. Many such annotations schemas have been proposed for different NLP applications. Annotation schemas are often designed to capture a wide range of linguistic informations. These include morphological, syntactic and semantic information. Some such well known annotation schemas include POS tagging, NE tagging, syntactic tree etc. All annotation schemas are developed towards solving a real world problem. In this paper, we aim towards arriving at an annotation scheme for cooking recipes.

Large number of cooking recipes are available in the web. People often follow these recipes while cooking. Annotation of cooking recipes can aid applications like complicated recipe search (e.g.,"find chicken pizza recipes that do not use mushrooms and can be baked in an electric oven"). It can also be used in recipe recommendation and adaptation. Our aim is to develop an annotation scheme for cooking recipes which can be used in above mentioned applications.

The rest of the paper is organized as follows. In Section 2, the related work in this domain is discussed. In Section 3, cooking recipes format is described briefly. Sections 4 and 5 discusses the features and structure of the proposed annotation scheme. In Section 6, completeness of the tagset is discussed. Section 7 describes the heuristics for automatically annotating the recipe text. Conclusions are drawn in Section 9.

2 Related Work

Many efforts have been done for extracting important information from recipes e.g. cooking ingredients, utensils, cooking actions, recipe recommendation (Teng et al., 2012), finding replaceable ingredients (Shidochi et al., 2009), recipe retrieval (Wang et al., 2008) etc. Ziqi et al. (2012) proposed different methods for automatically acquiring procedural knowledge in machine interpretable formats from natural language instructions like recipes. Shinsuke et al. (2014) used

* This work was done during the author's project work in IIT Guwahati.

flow graphs to annotate the recipes. These flow graphs are directed acyclic graphs in which vertex labels are named entities in the recipe, such as foods, tools, cooking actions etc and arc labels denote relationships among them. Valmi et al. (2012) proposed a semantic representation of cooking recipes using graphs. This representation is used for cooking recipes search and adaptation of new recipes according to user constraints. In this paper, we adopted a hierarchical and recursive schema for annotating the cooking recipes. Tagsets and recipe representation languages like Recipe Markup Language (RML),[1] RecipeBook XML[2] were defined earlier, but the primary goal of these representations is to allow people to create, store and share recipes in a variety of electronic formats and to convert from one format to another. Moreover, these tagsets capture linguistic information only at a shallow level and miss out the richer information present in the recipe text. Thus, we need a framework to describe the finer details of a cooking recipe text. The new framework proposed here addresses these deficiencies in an efficient and principled manner. The hierarchical and recursive schema enables us to capture various relations among the ingredients, devices/utensils and cooking actions and other attributes related to cooking action.

3 Cooking Recipes Format

Large number of recipe formats are available in the web describing the process of cooking. In general, there exist two main sections in a cooking recipe specification – ingredients list and procedure (directions). The *ingredients* section consists of various materials needed in preparing the dish and additional information like quantity, size, name of the items, preprocessing actions to be performed etc. *Procedure* includes various steps to be performed to prepare a particular dish. Each step contains cooking actions performed on the ingredients or intermediate materials formed from previous cooking steps, using utensils or devices. An example snippet of a cooking recipe taken from a popular website[3] is given below.

Ingredients:

1. *7 tablespoons olive oil, divided*
2. *Kosher salt, freshly ground pepper*
3. *2 garlic cloves, coarsely chopped*

Procedure: *Preheat oven to 350F. Toss bread and 3 tablespoons oil on a large rimmed baking sheet, squeezing bread so it absorbs oil evenly; season with salt and pepper. Spread out bread pieces in an even layer and bake, tossing occasionally, until crisp on the outside but still chewy in the center, 10 to 15 minutes. Let croutons cool.....*

4 Features of the Proposed Tagset

In this section we discuss the principles behind our tagset framework. The tagset proposed is hierarchical and recursive. Flat tagsets just list down the categories applicable for each unit of text without any provision for modularity or feature reusability (Baskaran et al., 2008). Hierarchical tagsets on the other hand are structured relative to one another and offer a well-defined mechanism for finding semantic relations between ingredients, utensils/devices, cooking actions and time.

4.1 Recursive and Hierarchical

The framework is recursive and forms a tree-structure. This ensures that instead of having a large number of independent categories, a recursive tagset contains a small number of broad categories at the top level, tagging larger units of text. Each broad category text has a number of subcategories in a tree-structure. The finer details of recipe text are captured in the separate layers of the hierarchy; beginning from the major categories in the top and gradually progressing down to cover specific features. This hierarchical arrangement helps in forming semantic relations between various elements of a recipe text.

5 Structure of the Tagset

In this section we describe our proposed framework where each node of the tree structure is explained. The tree starts with initial levels which describe the higher levels of the hierarchy.

5.1 Initial Levels

The framework has many layers organized in a tree structure. The initial levels cover broad categories. The root of the tree is the entire cooking

[1] http://www.formatdata.com/recipeml/spec/recipeml-spec.html

[2] http://www.happy-monkey.net/recipebook/

[3] http://www.epicurious.com

procedure, which is divided into ingredients section and procedure in the second level. The ingredients section contains descriptions of various ingredients used in the procedure. The procedure section is divided into number of individual steps. The details of the tagset are given in figure 1 and figure 2.

5.2 Ingredient Description

The subtree starts with ingredient description as the root. It describes Name of the Ingredient (NOI), Properties of Ingredients (POI), Total Size of the Ingredient (TSOI), Total Quantity of the Ingredient (TQOI), preprocessing actions performed on the Ingredients (IAOI). Properties (POI) is further divided into size, color, odor etc. Quantity (TQOI) is divided into number and units.

5.3 Step

Each step contains cooking actions performed on the ingredients or intermediate materials formed from previous steps, using utensils or devices. Time may also be specified in the step. Step is further divided into cooking actions, ingredients, utensils/devices, time specifications.

5.3.1 Cooking Action Specifications (ca)

Specifications are tagged under *ca* tag. They describe the verb of action, its purpose and how the action is to be performed.
(1) Stir the potatoes gently to make them soft.
Verb of action - *stir*
How the action is to be performed - *gently*
Purpose of action - *to make them soft*.

5.3.2 Utensils Specifications (uca)

Specifications of utensils or devices used in a cooking action like name, purpose, properties, usage are tagged under *uca* tag. The usage is specified using prepositions like *to*, *into*, *through*, *with*, *using*, *in*, *on*, *from* etc.
(1) through the mesh sieve, using the knife, from the bowl, into the jar.

5.3.3 Ingredients Specifications (ica)

Specifications of ingredients or materials used in a cooking action like name, quantity, size, properties are tagged under *ica* tag.
(1) Mix *3 tablespoons of kosher salt* with *potatoes* and stir the *mixture* using spatula.

5.3.4 Time Details (time)

Specifications of the time taken for performing a cooking action are tagged under *time* tag.
(1) Stir the eggs gently *until they turn into golden yellow color.*
(2) Stir *for 5 minutes.*

5.4 An Illustrative Example

An example of a recipe snippet annotated with the above defined framework is given below.
Snippet: 2 tablespoons , unsalted butter, divided

```
⟨ingredient description ingr id=''1''⟩
    ⟨TQOI⟩
        ⟨number⟩2⟨/number⟩
        ⟨units⟩tablespoons⟨/units⟩
    ⟨/TQOI⟩
    ⟨POI⟩
        ⟨state⟩unsalted⟨/state⟩
    ⟨/POI⟩
    ⟨NOI⟩butter⟨/NOI⟩
    ⟨IAOI⟩
        ⟨action⟩divided⟨/action⟩
    ⟨/IAOI⟩
⟨/ingredient description⟩
```

Snippet: Cook eggs, stirring gently about 3 minutes.

```
⟨step step-id=''4''⟩
    ⟨ca id = ''1''⟩
        ⟨action⟩Cook⟨/action⟩
    ⟨/ca⟩
    ⟨ica⟩
        ⟨NOI⟩eggs⟨/NOI⟩
    ⟨/ica⟩
    ⟨ca id = ''2''⟩
        ⟨action⟩stirring⟨/action⟩
        ⟨how-adv⟩gently⟨/how-adv⟩
    ⟨/ca⟩
    ⟨time⟩
        ⟨tnca⟩about
            ⟨number⟩3⟨/number⟩
            ⟨units⟩minutes⟨/units⟩
        ⟨/tnca⟩
    ⟨/time⟩
⟨/step⟩
```

6 Completeness of the Tagset

In this section, we discuss the completeness of our tagset. In general, an object is complete if nothing needs to be added to it. This notion is made more specific in various fields. In this context, it

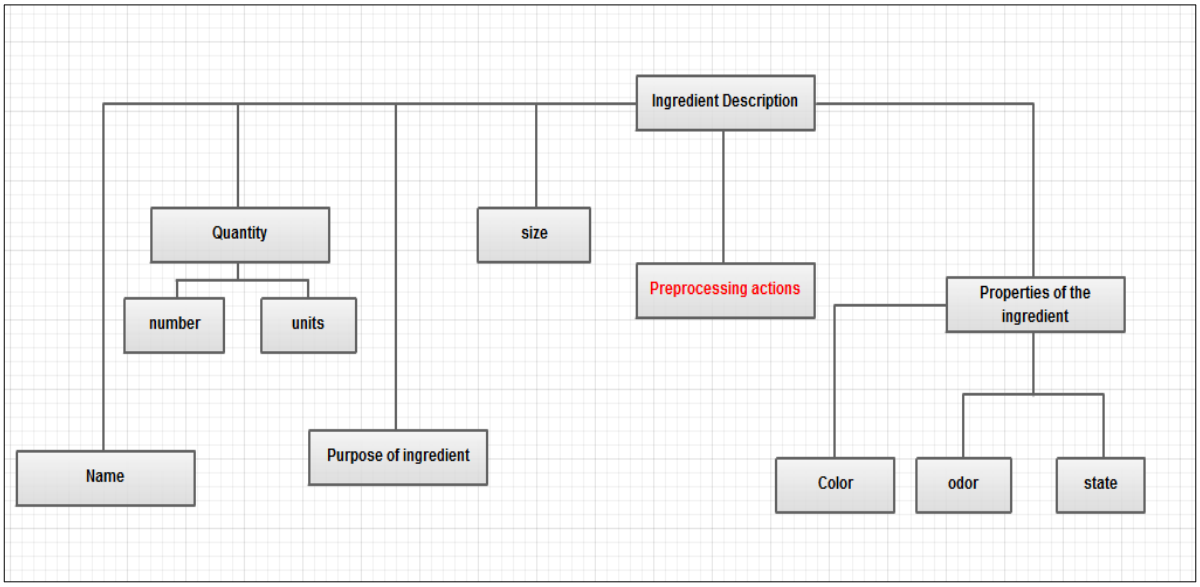

Figure 1: Ingredient Section. The nodes in red are not expanded further.

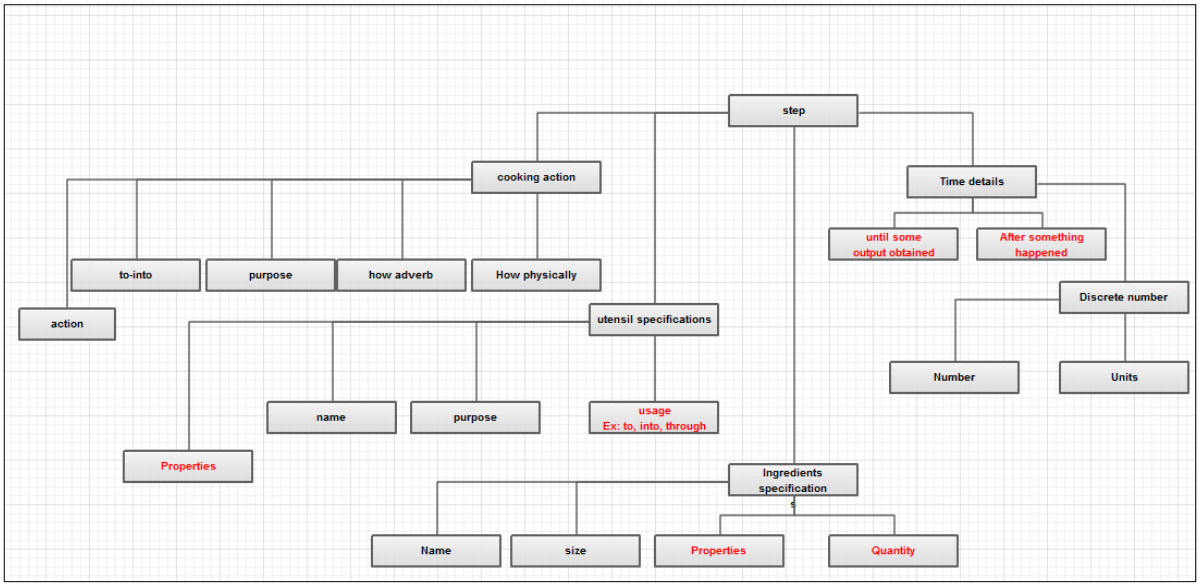

Figure 2: Procedure Section. The nodes in red are not expanded further.

means that, all the recipes can be annotated with the provided framework.

The smallest unit of a cooking procedure consists of performing a cooking action using utensils and ingredients to produce an output. The entire cooking procedure is the combination of these smallest units. Let us define this smallest unit as **atomic action.** Atomic action may include actions like stirring, cutting, heating, transferring etc. Each such action may require utensils or devices and materials or ingredients. Time taken for the atomic action may also be specified in the recipe text. We describe how these attributes are specified in a cooking recipe.

6.1 Specifications of Atomic action

Following are the different attributes under *cooking action*.

1. Main verb of action, which specifies action to be done
2. Purpose of the action
3. How the action is to be done

Following are the different attributes under *Ingredients* used in atomic action.

1. Name of the ingredient, size of the ingredient, quantity of the ingredient, properties of the ingredient

2. Direct ingredients - Taken from the Ingredients list

3. Intermediate ingredients - These are formed in previous steps as an intermediate product

Following are the different attributes under *utensils* used in atomic action.

1. Name of the utensil, size, properties

2. Usage of utensils is specified using prepositions like with, to, into etc

Following are the different attributes under *time* specifications used in atomic action.

1. Discrete values say, 5 minutes or

2. Until some desired output is obtained

3. Time is not specified or defined for some atomic actions

We have annotated and studied many recipes written in English language from a recipe website.[4] We found that all the above specifications of an atomic action can be tagged using the tagset defined earlier. Also the tagset framework is language independent and must be applicable to all languages as we are annotating semantic relations only.

7 Heuristics for Tagging the Recipe

We have defined heuristics to tag the input recipe based on the parts of speech of the words, parse structures of the sentences and lexical association measures. Heuristics for annotating ingredient section is based on the parts of speech(POS) of various tokens in the ingredient description. Parse trees are used for tagging procedure section. In addition, there are some bigrams which are to be tagged combinedly, like for example *olive oil*, *kosher salt* etc. We use a collocation measure, **salience score** (Pecina, 2010), as the lexical association measure for bigram modeling. The process of combining words into phrases and sentences of natural language is governed by a complex system of rules and constraints. In general, basic rules are given by syntax, however there are also other restrictions (semantic and pragmatic) that must be adhered to in order to produce correct, meaningful, and fluent utterances. These constraints form important linguistic and lexicographic phenomena generally denoted by the term collocation . Salience score is used to decide whether to tag bigrams combinedly or not. The salience score of a bigram is given by,

$$Salience(xy) = \log \frac{p\,(xy)^2}{p\,(x*)\,p\,(*y)} . \log f\,(xy)$$

where *p(xy)* is the probability of xy occuring together
f(xy) is the frequency of bigram xy
Here * means any token.

7.1 Ingredients Description

Usually ingredient section has a semi structured data. First, annotate the ingredient description using Stanford POS tagger[5] (Toutanova et al., 2003). Then scan through all the tokens formed from the input line in a linear fashion and annotate based on the heuristics. We describe some heuristics below. The heuristics given here are in a broad sense. Exact implementation details are omitted.

H1: Let k be the current token in the linear scan. If tag(k) is a **number** then annotate token k and $k+1$ as **quantity**. Inside this quantity tag, annotate k as **number** and $k+1$ as **units**.

H2: Let k be the current token in the linear scan. If tag(k) is a **adverb** and if it is followed by verb or noun or adjective, then annotate k and $k+1$ under **ca** tag. Inside the **ca**, annotate k as **how** and $k+1$ as **verb**.

H3: Let k be the current token in the linear scan. If tag(k) is a **adjective** and if tag($k+1$) is **noun** with salience(k, $k+1$) >threshold then annotate the bigram (k, $k+1$) as **NOI**.
Else annotate all the following adjectives under **properties** and next noun as **NOI**.

[4] http://www.epicurious.com

[5] http://nlp.stanford.edu/software/tagger.shtml

H4: Let k be the current token in the linear scan. If tag(k) is a **preposition**, such as *for* or *to* and if tag($k+1$) is **verb**, then annotate the pair (k, $k+1$) under purpose of ingredient tag.

Below is the sample output obtained using above heuristics.

⟨TQOI⟩
⟨number⟩2⟨/number⟩
⟨units⟩tablespoons⟨/units⟩
⟨/TQOI⟩
⟨POI⟩
⟨p1⟩golden⟨/p1⟩
⟨/POI⟩
⟨NOI⟩butter⟨/NOI⟩
⟨POI⟩
⟨color⟩brown⟨/color⟩
⟨/POI⟩
⟨NOI⟩flaxseeds⟨/NOI⟩

7.2 Procedure

The procedure section consists of natural language text. We used tools like natural language parser and WordNet (2010) in the process of annotating the recipe. A natural language parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. Parse tree gives a complete picture of relations between various phrases in the sentence. These relations can be used to form semantic relations in our analysis, which will be helpful in annotating the recipe. Stanford Parser[6] (Socher et al., 2013), a statistical parser is used for our purpose. WordNet gives the sense of various words in the recipe text. For example, if we query WordNet with the word *avocado*, the output contains *noun.food*. This information can be used for classifying words into utensils, ingredients and cooking actions etc.

The first step involves the tokenization of procedure into many individual sentences. Each sentence is annotated as *step*. Each step is then parsed and the parse tree is analyzed to annotate various details inside that step.

7.2.1 Handling Various Phrases

The most common phrases to be handled includes noun phrases (NP), verb phrases (VP), adjective phrases (ADJP), adverb phrases (ADVP), prepositional phrases (PP) and some intermediate nodes of the parse tree like S (sentence), SBAR (Subordinate clause), WHADVP (Wh-adverb phrase) etc. These phrases and nodes are handled recursively. The recursion stops at the leaves of the tree. The modules for handling NP, VP, PP, ADVP, ADJP are described below. Below is the parse tree for the sentence "Add crushed black peppercorns and lemon juice and mix well".

```
(ROOT
    (S
      (VP
        (VP (VB Add)
          (NP
            (NP (JJ crushed) (JJ black) (NNS peppercorns))
            (CC and)
            (NP (JJ lemon) (NN juice))))
        (CC and)
        (VP (VB mix)
          (ADVP (RB well))))
      (. .)))
```

Noun phrases:
A noun phrase can consist of one word (for example, the pronoun *it* or the plural noun *apricots*), or it can consist of a noun with a number of dependents. The dependents occur before or after the noun head depending on their function. All noun phrases (NPs) have a noun or pronoun as the head. The noun is the anchor of the phrase and the phrase will not be grammatical without it. Noun phrases often function as complements to the verb.
In the NP, *crushed black peppercorns*, peppercorns is the head word (Noun) and crushed, black are dependants (adjectives). This NP is the complement of the verb *Add*, which is the cooking action. First we find the sense of the head word and depending upon the sense we carry out further analysis. For example, if sense shows it is an ingredient then annotate as name of the ingredient(noi), its properties (poi), quantity and units in that NP. Similarly other cases are covered.

Prepositional Phrase:
A prepositional phrase has a preposition as its head and this is usually followed by a noun phrase. A PP relates the dependent NP to other constituents in a sentence in terms of place (for example, *in the flour*).

[6] http://nlp.stanford.edu/software/lex-parser.shtml

Verb Phrase:
Each verb phrase contains one main verb. This is most probably the cooking action. Each VP contains only one main verb that functions as the head of the VP. Eg: *Add* is the head of the VP given in above example. The main verb is tagged as cooking action and the surrounding adverb phrases and adjective phrases tells how the action is performed.

Adverb Phrase:
An adverb phrase takes an adverb as its head. **AdvP**s typically modify a verb within a VP (for example, Mix *well*). They usually describe how the cooking action is to be performed.

Adjective Phrase:
An adjective phrase has an adjective as its head. An AdjP can function as a modifier within a noun phrase (for example, the *brown* flaxseeds) in which case it is annotated under property.
The following are some issues which are taken care while automatically tagging the input recipe.

1. **Co-reference**: In linguistics, co-reference occurs when two or more expressions in a text refer to the same person or thing. Ingredients or intermediate outputs and utensils from the previous steps are co-referenced in the current step. Usually pronouns refer to nouns. We are using Stanford Deterministic Coreference Resolution System[7] (Lee et al., 2013) to resolve the coreferences.

2. **Distance references**: The tags corresponding to a particular subtree may not be at consecutive positions in the input recipe. So for an individual tag, if it is not inside its subtree, we have to keep a reference to its parent using an attribute called *ref*.
   In the example below, *action*, *how-adv* have same parent *ca*. But they are not present in consecutive positions, so *how-adv* makes a reference to *ca* using *ref* attribute.
   Cut potatoes smoothly using knife

```
⟨step step-id=``4''⟩
   ⟨ca id = ``1''⟩
      ⟨action⟩Cut⟨/action⟩
   ⟨/ca⟩
   ⟨ica⟩
      ⟨NOI⟩potatoes⟨/NOI⟩
   ⟨/ica⟩
      ⟨how-adv
ref=``ca-1''⟩smoothly⟨/how-adv⟩
   ⟨uca⟩
      ⟨using-uca⟩using⟨/using-uca⟩

      ⟨nuca⟩knife⟨/nuca⟩
   ⟨/uca⟩
⟨/step⟩
```

8 Results

The above heuristics are tested on over 100 recipes collected from the web and results are compared with manually annotated recipes. Results on only 20 recipes are shown below. These recipes are collected from an Indian website[8] and a foreign website.[9] The evaluation is done as follows.

1. **Scheme A:** Each step in the procedure can get a maximum score of 1. This maximum score is divided equally among various *ca, ica, uca, time* tags present in that step. For example, if a step has 2 *ca* tags, 1 *uca* tag, 2 *ica* tags and 1 *time* tag, the maximum score each tag can get is $1/6$, as there are 6 of them present in that step. The score is further divided among their children. In addition to dividing score among the children, score is also divided for coreferencing the pronouns. Then evaluation is done at each tag. If the annotation is correct, it gets the maximum score it was assigned. The score of the step is calculated by adding scores of individual tags in that step. Once the score for each step is calculated, mean of scores of all the steps in the recipe is calculated. This mean score is called the *score of the recipe*. Average of the scores of all the tested recipes gives the accuracy of the system.

2. **Scheme B:** Some errors in the annotated output are due to the wrong output of the Stanford parser. In this scheme we want to find out how much gain in accuracy can be obtained if no parser error exist. So if an error occurs in annotation because of the wrong

[7] http://nlp.stanford.edu/software/dcoref.shtml

[8] http://www.sanjeevkapoor.com/

[9] http://www.epicurious.com/

parse tree of the sentence, that part of the annotation is not considered while scoring the sentence i.e, score is not divided among those tags.

Table 1: Results of evaluation of some recipes

Recipe Number	Scheme A	Scheme B
1	0.63	0.9
2	0.74	0.98
3	0.78	0.83
4	0.85	0.9
5	0.81	0.96
6	0.89	0.96
7	0.72	0.75
8	0.82	0.94
9	0.82	0.95
10	0.71	0.94
11	0.65	0.98
12	0.74	0.94
13	0.77	0.94
14	0.75	0.85
15	0.77	0.90
16	0.68	0.85
17	0.81	0.98
18	0.72	0.86
19	0.72	0.95
20	0.72	0.94
Avg. Accuracy	0.75	0.91

8.1 Inferences

Given a step in a procedure, the accuracy scores in Table 1 signifies that the step can be annotated with an accuracy of 75% in scheme A evaluation and an accuracy of 91% in scheme B evaluation. We observe in bar graph of Figure 3 that scheme A scores are less than Scheme B scores. This infers that if the accuracy of the Stanford parser is increased we can annotate the recipe with greater accuracy.

9 Conclusion and Outlook

In this paper we have presented a tagset framework designed for cooking recipes and a system which automates the recipe annotation process using the proposed framework. The hierarchical recursive tagset framework captures relations among different attributes through linguistic analysis. Furthermore, we have shown that the automatic annotation with out tagset framework achieves reasonably good accuracy using language tools and heuristics.

The heuristics used for the current system required language specific analysis which may not be available for many languages. In order to make the system language independent, we plan to use machine learning-based approach to identify different attributes and relationship from the recipe text for automatic annotation.

References

Sankaran Baskaran and Kalika Bali and Tanmoy Bhattacharya and Pushpak Bhattacharyya and Girish Nath Jha and Rajendran S and Saravanan K and Sobha L and Kvs Subbarao. 2008. *A Common Parts-of-Speech Tagset Framework for Indian Languages* In Proc. of LREC 2008.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4), pages. 885-916,

Shinsuke Mori and Hirokuni Maeta and Yoko Yamakata and Tetsuro Sasada. 2014. Flow Graph Corpus from Recipe Texts, In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, *European Language Resources Association (ELRA)*.

Pavel Pecina. 2010. Lexical association measures and collocation extraction *Language resources and evaluation*, Vol.44, pages. 137-158, Springer.

Princeton University "About WordNet". WordNet. Princeton University. 2010. ⟨http://wordnet.princeton.edu⟩

Yuka Shidochi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase. 2009. Finding Replaceable Materials in Cooking Recipe Texts Considering Characteristic Cooking Actions. In *Proceedings of the ACM Multimedia 2009 Workshop on Multimedia for Cooking and Eating Activities*, ACM, pages. 9-14.

Richard Socher, John Bauer, Christopher D. Manning and Andrew Y. Ng 2013. Parsing With Compositional Vector Grammars, In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, pages. 455-465.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network In *Proceedings of HLT-NAACL 2003*, pp. 252-259.

Chun-Yuen Teng, Yu-Ru Lin, and Lada A. Adamic. 2012. Recipe Recommendation Using Ingredient

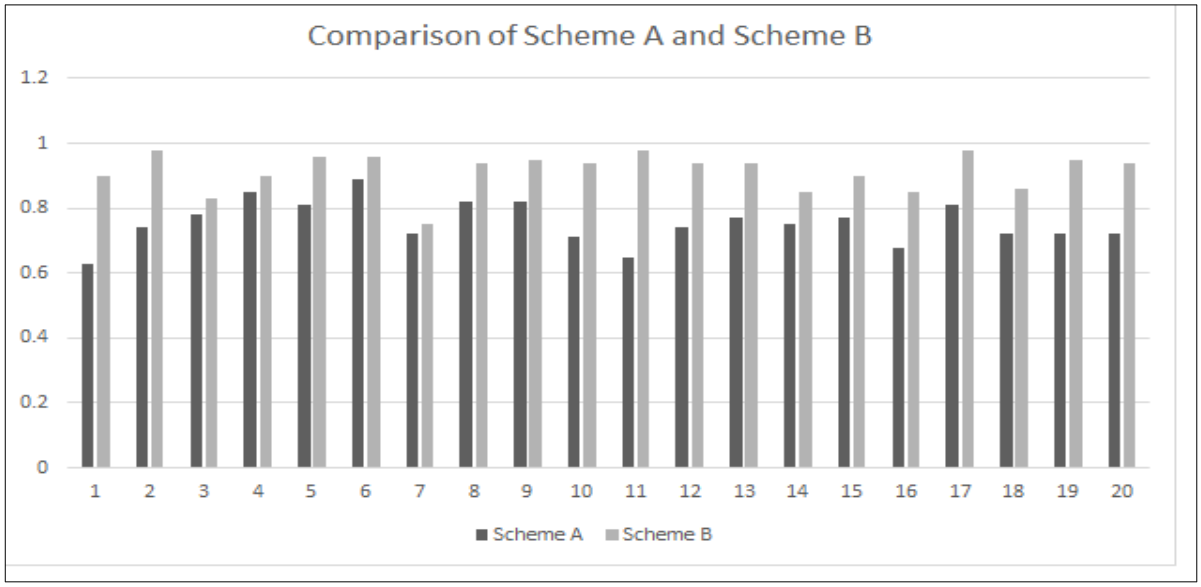

Figure 3: Comparison of results in Scheme A and Scheme B.

Networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, ACM, pages. 298-307.

Dufour-Lussier Valmi, Le Ber Florence, Lieber Jean, Meilender Thomas and Nauer Emmanuel. 2012. Semi-automatic annotation process for procedural texts: An application on cooking recipes.

Liping Wang, Qing Li, Na Li, Guozhu Dong, and Yu Yang. 2008. Substructure Similarity Measurement in Chinese Recipes. In *Proceedings of the 17th International Conference on World Wide Web*, ACM, pages. 979-988.

Ziqi Zhang and Philip Webster and Victoria Uren and Andrea Varga and Fabio Ciravegna. 2012. Automatically Extracting Procedural Knowledge from Instructional Texts using Natural Language Processing. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.