Recipe Pos Tagger*

Mehmet Özgen
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
mehmet.ozgen@hacettepe.com

İbrahim Ardıç
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
ibrahim.ardic@hacettepe.edu.tr

Gönenç Ercan
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
gonenc.ercan@hacettepe.edu.tr

Pınar Duygulu Şahin
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
pduygulu@hacettepe.edu.tr

ABSTRACT

A set of instructions is needed to systematically determine the steps of a task. The accuracy of this task is usually understood as the result of the work done. In this context, the correctness of instruction to action mapping is important in the recipes to be done consistently. In this work, we proposed a model by learning with the language processing techniques (Word2Vect, CRF++) of the work steps specified in the field which is specific to the field to reveal action order in the unclear situations. Furthermore, we build a parser that extract actions, ingredients and tools from given recipes and generate an action graph from parser results which accuracy is about 76 percent.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

Information Retrieval, Natural Language Processing, weaky-labelled data, text analysis, big data

ACM Reference Format:
Mehmet Özgen, İbrahim Ardıç, Gönenç Ercan, and Pınar Duygulu Şahin. 1997. Recipe Pos Tagger. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97),* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Part of speech (POS) tagging is the key topic for information retrieval area to extract the valuable information that is used for classification, clustering etc. from any natural languages.

In this paper, it is aimed to automatically model the steps of a certain task by using text data which is easily accessible from the internet and to create methods and tool that can help people since there are many recipe documents and videos available on the internet, in this paper, working area is selected as cooking recipes. POS tagging is important that it constructs a description a set of instructions of a recipes which forms content based text analysis of our work.

In this paper we describe a rule-based tagger which performs as well as taggers based upon probabilistic models. The rule-based tagger overcomes the limitations common in rule-based approaches to language processing: it is robust, and the rules are automatically acquired. In addition, the tagger has many advantages over stochastic taggers, including: a vast reduction in stored information required, the perspicuity of a small set of meaningful rules as opposed to the large tables of statistics needed for stochastic taggers, ease of finding and implementing improvements to the tagger, and better portability from one tag set or corpus genre to another.

2 RELATED WORKS

Algorithmic processing of training texts and / or commands expressed in natural language has long been an interesting question (Harnad, 1990). In this problem, the artificial intelligence agent (such as a robot) is trying to learn the operations corresponding to symbols that are expressed in a field. The system is expected to automatically interact with definitions in the physical world as symbols to interact with objects and objects. We obtained through www.allrecipes.com

Chen and Mooney (2011) have developed a system that can act according to the expressions expressed in natural language in the system they developed. Based on the reinforced learning methods, the system can translate the sentences into natural language by using the information in the location (such as the objects on the wall) and the state flow that is modeled. As can be seen from these studies in the literature, reinforcement learning techniques are used in problems where the current situation can be modeled more clearly. A slightly different problem is Kushman et al. (2014) developed a similar method for solving problems expressed in mathematical texts. According to this method, the system is mapping the problem described in the text to a set of equations and producing the result. To learn the method, we use a training set marked at different levels. Equations that can be created in the system can be given or can be learned only when the last answer is given. In the method they

*Produces the permission block, and copyright information

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

developed, the solution is defined as diagrams, and a probability model that maps the information given in the text to the diagram is developed. Of course, it is necessary that the structure of the questionnaire is similar and that all of them can be transferred to the equation diagram.

Recipes can also be thought of as a set of steps that must be followed. From this point of view, there are common aspects to the problems described above. The most important difference is that it is difficult to convert the media knowledge to a reward function to provide reinforcement learning. The biggest reason for this is that it is difficult to model the correct sequence of operations that can be done while cooking. To overcome this problem in the processing of recipes, three different basic approaches are striking. The first is knowledge-based methods that are dependent on previously created knowledge vocabulary, the second is those using supervised learning methods, and the third is unsupervised learning and extracting a specific model from the data in the compilation. The first approach involves all the recipes, contents, tools and processes, and methods that can show the relationship between them. In the second approach, it is necessary to generate the data with the equivalent of the text for the training compilation. In the third category, transductive learning is performed according to the solutions given, ie it is not possible to classify the given samples and to process new ones.

2.1 Information Based Methods

Walter et al. (2011) describes a method of generating flow diagrams from recipe texts, extracting recipe statements as material, processing and finishing conditions. Based on this information, a flow diagram is created. This method requires pre-tagged sentences and a dictionary that holds all of these tags.

Another problem that may be related is the creation of an information dictionary. Gaiilard et al. (2012) foods and their relationships with a Wiki-style method. Its approach basically consists of 6 hierarchies; diet and mealtime according to different features categorize the food. With this approach it is possible to adapt a new recipe or an existing recipe.

Although food-related ontology is increasingly detailed, these methods often fail to adapt to dynamic language and variety. Thus, when ontology methods are used alone, successful results are achieved for some data sets, but success is often low. Moreover, in order to adapt these methods to a different language, ontologies should be structured in accordance with other languages.

2.2 Supervised Learning Methods

One of the most important features of tutorial learning techniques is the target description model. Basically, the information that this model needs to be able to show is where the processes described in the steps are done on which materials and where the output of this step is used. Some of these associations show in the tree structure (Jermsurawong and Habash, 2015), while some researchers show it in a non-cyclic manner (Malmoud et al., 2014). It is possible that this notation can both express all the recipes and learn it automatically. Jermsurawong and Habash (2015) defined a representation in the tree data structure that stores food items and their relationships over the description steps. This demonstration can show which step of the recipes is related to which steps and materials.

Malmoud et al. (2014) sees text in recipes as a Markov Decision Process problem by expanding semantic role labeling. It holds two things between the process and the material. The purpose of this relationship is to demonstrate the processes within the training that cause these situations to occur. Mori et al. (2014) created a labeling tool by showing recipes as non-cyclic charts in their initial work. With this tool, the recipe created a graph that indicates the skis over the text. The given Japanese recipes first extract the word segmentation, locate the words in the clan's tasks, label the named entities, and finally construct the predicate-argument structure. In order to resolve the uncertainties in the contents, Erica Greene (2016) tagged a total of 187,000 sentences for labeling and edited the tags of the words in the table of contents using the Conditional Random Fields method.

2.3 Unsupervised Learning Methods

In 2015, Kiddon et al. (2015) has developed a method of learning without a teacher for the processing of recipes. Unlike previous works, this method learns the linkages of the line by using different parameters according to the variables in the system and using the expectation maximization method. Since this model is constructed according to the definition in the drawing, it is not possible to learn the features of an extragalactic recipe. In this respect, it is a transduction method, all the recipes to be processed have to be obtained at the model learning time. Moreover, the generated data model has a high number of implicit variables because it defines both the verbs, the contents, and the content of the contents in relation to the steps as one of the parameters of the model. When such high numbered models are learned, they can be found in a high number of local maxima, which prevents the optimum model from being found. It is intended to remove them as to which action is associated with the material, removing the flow of the recipe as a whole. These methods are inspired by semantic spaces used in text mining and meaning spaces that show word meaning relations. By using these material spaces, it is possible to determine which materials can be replaced instead of which materials. Nedovic (2013) used a similar method to define a method of learning materials for different types of food. Latent Dirichlet Allocation (LDA) and Deep Belief Networks (DBN) are used in the method. It has been seen that the output of the system can group materials according to the foods in different kitchens. A similar study has been proposed by Achananuparp and Weber (2016) to produce safer food recommendations in meals. In this method, Singular Value Decomposition (SVD) method, which is widely used for extracting word meaning relations, is used.

3 EXPERIMENTAL SETUP

As seen in the literature, there are many methods aiming at revealing the relationship between the description texts and the contents of the description texts, the sequence of the process flow and the situations that occur during the actions and showing them with the models by many different methods. The texts should be easily perceivable by the computer by passing about 18000 recipes we obtained through www.alrecipes.com through certain operations.

Using NLTK libraries, each of the recipes was first converted into a clean text by separating individual recipes and separating them into words, punctuation, and meaningless words. (names, categories, contents, descriptions, and comments). Then each recipe is divided into sentences for labelling.

Algorithm 1: Data Retrieval and Preprocessing Overview

```
Function ReadProcessData CvsFiles data
    forall eachRecipe c in data do
        (ingredients, direction) = readDataFromCSVfile(c)
        ingreNewTag = calculateTAGWithCRF(ingredients)
        tokenizedD = tokenizeSentence(direction)
        taggedD = calculateTagsWithPOSTag(tokenizedD)
        taggedDirection = updateTag(taggedD, ingreNewTag)
```

This section we prepare data for the process, firstly, read data from csv file which is generated before and divide data into two parts, direction and ingredients. Using the NLTK library, labeling was done according to the culled state (VB, NN, ADJ, etc.) that each of the belts passed. As a result of the labeling made, "I can chopped green chile peppers" in the table of contents is labeled as below in TABLE 1. It was observed that 450 of 171.224 labeled samples were separated and tested and 89 percentage correctly tagged.

As can be seen in Table 1, it does not allow me to know that the word "life" is a unit of measure, "1" actually tells the quantity, and that the words "green" and "choped" are actually interpretations of a "papers" word.

The challenge of parsing the recipe is to be able to distinguish content components from component cues. Erica Greene (2016) has trained 171,244 sets (labeled as UNIT, QUANTITY, COMMENT and OTHER) with the very specific set of CRF ++ (Conditional Random Fields) method she has created to solve this problem and has been able to label the contents portion probabilistically with a newly given description. Let us have the sentence "1 teaspoon sugar". The model is using 171,224 tagged data to learn a model that can predict the tag sequence for any sentence we have given to it, even though we have never seen this component count before. It approaches this by modeling the conditional probability of a set of labels.

p (UNIT UNIT UNIT| "1 teaspoon sugar")
p (QUANTITY UNIT UNIT| "1 teaspoon sugar")
p (UNIT QUANTITY UNIT| "1 teaspoon sugar")
p (UNIT UNIT QUANTITY "1 teaspoon sugar")
p (UNIT QUANTITY QUANTITY "1 teaspoon sugar")
p (QUANTITY QUANTITY QUANTITY | "1 teaspoon sugar")
p (UNIT QUANTITY NAME| "1 teaspoon sugar") Secondly, tag direction and ingredients wtih the help of NLTK and CRF++ libraries. Tagging direction according to NLTK libraries is not satisfied in order to generate action graph. For this reason, update and change tags after NLTK process is finished.

Table 1: Tags

chop	green	chile	peppers
VB	NN	NN	NN

Algorithm 2: Pos Tagging Overview

```
Function updateTags Direction taggedDirection
    forall sentence in taggedDirection do
        updateVerbs(sentence)
        updateTools(sentence)
        updateIngredients(sentence)
    findRelatedVerbsPair(direction)
```

Table 2: Tags

chop	green	chile	peppers
VB	CMMT	NAME	NAME

Table 3: Tags

put	the	peppers	into	the	pan
VB	DET	NAME	ADP	DET	TOOLS

Table 4: Tags

put	the peppers	into the pan
PRED	INGEDIENTS	NON IGRE SPAN

As mentioned above, it calculates all the probabilities that can be labeled "1 teaspoon sugar". The beauty of the linear-chain CRF model makes some conditional independence assumptions that allow us to use dynamic programming to efficiently search the area of all possible label sequences. As a result, we have re-tagged our data with Erica Greene (2016), and the result is shown in TABLE 2. The results are shown in Table 2, which shows the best label sequence at a time that is linear with the number of second.

After, tags are updated according to neighbor tags, if the tag is "VB" and the neighbor tag is "DET" (determiners), two words are concatenated by the algorithm and tagged by "PRED" . The CRF++ algorithm is tagged each word separately but the same tag, these are concatenated. That is depicted on the TABLE 3 and TABLE 4.

Finding action pairs with proximity values are required for graph generation. Generally, actions are sequential in the recipes, however, actions can be done parallel or some actions take duration while doing. So that, "preheat- PRED" is related with "bake for- PRED". This results show the relation between "preheat- PRED" and "bake for- PRED". In order to link "preheat- PRED" and "bake for- PRED" in the graph, we look at the similarity of the cosine with the actions.

3.1 Calculation of Proximity

Kiddon et al. (2015), some words are not in the table of contents, but more than one is meant. (Eg mixture, them) Kiddon et al. (2015) defines this as a hidden object. He has created a probabilistic model to make hidden objects clear. In order to reveal these hidden objects, we present a relationship between the words 'NAME' in this action and its contents. In fact, each word is represented as a vector and we calculate the cosine similarity between them.

Using the Word2Vec library that displays 3.5 billion words created by a working group led by Tomas Mikolov (Google) as a 300-dimensional vector vocabulary, the words that pass through the intellectual part and are labeled 'NAME' have been converted into a 300-dimensional vector.

In the recipe sentence set, phrases without any word labeled 'NAME' are removed. When the words labeled as 'PRED' are extracted, all the words are converted into 300 dimensional vectors and the averages are taken. Because the other words around the word tagged by "VB" can also provide us with information about what materials the hidden object contains. If you only go through the actions, you will be made a comment by looking at the similarity of only two calves. However, the sentence can sometimes contain words that characterize the words in the contents. Let's take a look at "Cook the mixture until caramelized". It is labeled as "cook-PRED". When we look at the similarity of the cosine with the contents, it is seen that almost all of them resemble to each other. However, when the complex average of sentence is taken, "caramelized" word is included. And the result is more similar to "Onion" and "sugar". Taking the average of the blame for this reason actually brings us closer to the right conclusion. On the other hand, actions are generally sequential, some actions related with other actions. "preheat" is the action which is related with "bake for" action. Because after "preheat the oven" , "bake for the mixture". this "preheat" action should be before the "bake for" action. "preheat" action result is related with "bake for" action. In order to find this relation and similarity about whole direction in the direction, we use Word2Vec library for finding cosine similarity between the actions. If the similarity bigger that threshold value that we select, the pairs and the similarity value are held in the list. Then the list is used for linking the action nodes in the graph.

Algorithm 3: Graph Generation and Validation

```
Function generateGraph taggedRecipe recipe, verbPairs, pairs
    actionNodeList =[]
    forall sentence in recipe do
        actionN = generateSentNode(sentence)
        actionNodeList.append(actionN)
    forall action in actionNodeList do
        linkTheAction(action, pairs)
```

3.1.1 Inline (In-text) Equations. A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [**?**]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

3.1.2 Display Equations. A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

3.2 Citations

Citations to articles [**? ? ? ?**], conference proceedings [**?**] or maybe books [**? ?**] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [**?**]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide* by Lamport.

This article shows only the plainest form of the citation command, using `\cite`.

Some examples. A paginated journal article [**?**], an enumerated journal article [**?**], a reference to an entire issue [**?**], a monograph (whole book) [**?**], a monograph/whole book in a series (see 2a in spec. document) [**?**], a divisible-book such as an anthology or compilation [**?**] followed by the same example, however we only output the series if the volume number is given [**?**] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [**?**], a chapter in a divisible book in a series [**?**], a multi-volume work as book [**?**], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [**?**], a proceedings article with all possible elements [**?**], an example of an enumerated proceedings article [**?**], an informally published work [**?**], a doctoral dissertation [**?**], a master's thesis: [**?**], an online document / world wide web resource [**? ? ?**], a video game (Case 1) [**?**] and (Case 2) [**?**] and [**?**] and (Case 3) a patent [**?**], work accepted for publication [**?**], 'YYYYb'-test for prolific author [**?**] and [**?**]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [**?**]. Boris / Barbara Beeton: multi-volume works as books [**?**] and [**?**].

A couple of citations with DOIs: [**? ?**].

Online citations: [**? ? ?**].

Table 5: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Figure 1: A sample black and white graphic.

3.3 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 5 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 6 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

It is strongly recommended to use the package booktabs [**?**] and follow its main principles of typography with respect to tables:

(1) Never, ever use vertical rules.
(2) Never use double rules.

It is also a good idea not to overuse horizontal rules.

3.4 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of `.eps` files to be displayable with LaTeX. If you work with pdfLaTeX, use files in the `.pdf` format. Note that most modern TeX systems will convert `.eps` to `.pdf` for you on the fly. More details on each of these are found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. And don't forget to end the environment with **figure***, not **figure**!

Figure 2: A sample black and white graphic that has been resized with the `includegraphics` command.

3.5 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. ACM uses two types of these constructs: theorem-like and definition-like.

Here is a theorem:

Theorem 3.1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)\,dt = G(b) - G(a).$$

Here is a definition:

Definition 3.2. If z is irrational, then by e^z we mean the unique number that has logarithm z:

$$\log e^z = z.$$

The pre-defined theorem-like constructs are **theorem**, **conjecture**, **proposition**, **lemma** and **corollary**. The pre-defined definition-like constructs are **example** and **definition**. You can add your own constructs using the *amsthm* interface [**?**]. The styles used in the `\theoremstyle` command are **acmplain** and **acmdefinition**.

Another construct is **proof**, for example,

Proof. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c}\left[gx \cdot \frac{f(x)}{g(x)}\right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. □

4 CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

Table 6: Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

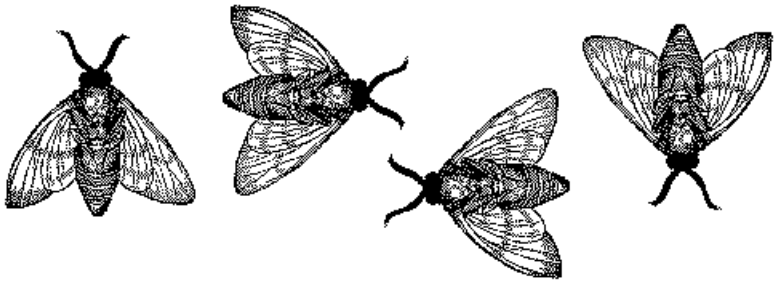

Figure 3: A sample black and white graphic that needs to span two columns of text.

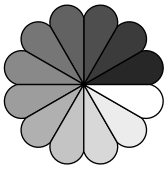

Figure 4: A sample black and white graphic that has been resized with the `includegraphics` command.

A HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e., the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters.

A.2.2 Math Equations.

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations.

A.2.4 Tables.

A.2.5 Figures.

A.2.6 Theorem-like Constructs.

A Caveat for the TEX Expert.

A.3 Conclusions

A.4 References

Generated by bibtex from your `.bib` file. Run latex, then bibtex, then latex twice (to resolve references) to create the `.bbl` file. Insert that `.bbl` file into the `.tex` source file and comment out the command `\thebibliography`.

B MORE HELP FOR THE HARDY

Of course, reading the source code is always useful. The file `acmart.pdf` contains both the user guide and the commented code.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Yuhua Li for providing the MATLAB code of the *BEPS* method.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304 and Young Scientists' Support Program (http://www.nnsf.cn/youngscientists).