A PATTERN BASED APPROACH FOR THE DERIVATION OF BASE FORMS OF VERBS FROM PARTICIPLES AND TENSES FOR FLEXIBLE NLP

Ram Gopal Raj and S. Abdul-Kareem

Department of Artificial Intelligence
Faculty of Computer Science and Information Technology
University of Malaya, 50603 Kuala Lumpur
Email: gopalraj@tm.net.my, sameem@um.edu.my

ABSTRACT

Natural Language Processing (NLP) is an integral part of a conversation and by proxy an integral part of a chatterbot. Building a complete vocabulary for a chatterbot is a prohibitively time and effort intensive endeavor and thus makes a learning chatterbot a much more efficient alternative. Learning can be performed from many facets including individual words to phrases and concepts. From the perspective of words, the grammatical parts of speech become important since they allow meaning and structure to be derived from a sentence. Verbs tend to be unique since they have different forms, namely participles and tenses. As such we present an algorithm to derive the base verb from any participle or tense.

Keywords: tense, participle, algorithm, pattern

1.0 INTRODUCTION

During the development of our self-learning tele-text conversational entity, or chatterbot, called RONE, we needed to develop suitable knowledge representation scheme and the appropriate sentence dissemination methods, [1]. Of course, since chatterbots are intended to converse with humans, a suitable form of Natural Language Processing (NLP) needed to be implemented. Other research in NLP as well as machine learning indicates that it is possible to create lexical classes from data sets with some accuracy, [2], [3] and [4]. Lexical classifications are a technique that accommodates certain critical NLP functions such as word sense disambiguation and parsing which are important in answering questions and information matching, [5], [6], [7], [8] and [9], all of which are critical to chatterbots. Lexical classes can be used to form generalizations and take into account the syntax of a sentence as formed by its individual words, [10] and [11], as opposed to purely semantic classes, [12]. When mentioning syntax, parsers and subcategorization represent deep syntactic features, [13], [3] and [4], and chunkers and taggers represent shallow syntactic features, [2]. However, most lexical classes have to be painstakingly defined manually during system development and as such are rarely very comprehensive. As such, increasing emphasis has been placed on automatic classification and since verbs are generally the main predicates in sentences, verbs are the focus of such systems, [2]. An example of an automatically built lexicon is VALEX, [14].

When getting RONE to perform NLP, we found that it is more efficient to allow our system to add to its own vocabulary instead of building in everything all at once. We also found that utilization of sentence grammar helps the system "understand" a sentence better. When, mentioning vocabulary, and grammar, a system needs to know the parts of speech that each word in the system's vocabulary belongs. As such learning new words requires the derivation or prediction of the part of speech that each new word belongs to. Since conversational ability is reliant on NLP, it would be logical to conclude that advances in conversational system technology would be potentially beneficial to NLP. The importance of verbs in NLP is demonstrated in its usefulness when used in establishing relationships between nouns, [15]. RONE utilizes subject object verb relationships to form an understanding of what the user has said. A natural language text analyzing algorithm was presented by [16]. Part of the algorithm utilized the subject object verb relations of text to extract object oriented modeling elements for their study, but clearly showed the importance and usefulness of such relations.

1.1 Part of Speech Derivation

Since the system, [17], was developed based on the rules of English Language Grammar, the next step after looking at individual words is to match those individual words to their respective parts of speeches. The individual parts of speech are stored in the SQL database that is the knowledge base component of the system. Unless we input a ridiculous number of words and their parts of speech into the knowledge base, there is no doubt that RONE will encounter a word that will not return a match from the knowledge base. Since RONE is built to learn, he must also learn form multiple facets, new words, new information and etcetera. When a word does not turn up a part of speech

match, then RONE utilizes a prediction algorithm. The prediction algorithm is based on patterns we determined in sentences. The patterns involve the sentence structure in terms of the parts of speech of the words leading up to and the word after the unknown word. A simple illustration of a pattern for predicting the part of speech of an unknown word is that if the word preceding the unknown word is 'a', 'the', or 'an' then the unknown word is without a doubt, either and adjective or some form of noun, be it a pronoun or otherwise. These rules were developed based on observations of sentence structures.

1.2 The Issue of New Verbs

When a new word is found to be a verb, it can pose a problem. Since RONE is tense specific and also recognizes the differences between basic verbs and their other forms such as participles, any new verbs must be decomposed into its basic form if it is not already as such, and then expanded into all the different tense and participle forms respectively for storage. For example, if the previously unknown word 'pulled' is encountered, it is identified as a verb by the 'unknown word prediction algorithm'. It must then be turned into the basic verb 'pull' and stored with its past tense 'pulled' and participles 'pulling' and 'pulls'.

We noted during our study, that verbs fall into certain patterns. Unfortunately the patterns though distinct were not fully consistent. Therefore the rules for the algorithms of decomposing the verbs needed to utilize general rules with multiple exceptions for the verbs that did not fall within the patterns.

1.3 Verb and the Determining the Tense of a Sentence

The verbs are processed for a match in the relevant past, present and future denotations. We have found that the key verb is always the first verb in a phrase, meaning that the tense of a sentence is always given away by the initial verb. Take for example the following sentences:
"Harry **ran** away from school." = Past tense
"**Did** Harry run away form school?" = Past tense
"Harry **is** the youngest boy in his class." = Present tense
"What **is** today's date?" = Present tense
"Harry **will** take his exam tomorrow." = Future tense
"What **will** Harry do tomorrow?" = Future tense

The initial verb for each of the sentences is in bold. Notice that the verb mentioned for each sentence is in the tense that the sentence is in regardless of all the following verbs. The exception to this rule is that should a time related noun be included then the tense of the sentence can be affected, for example, "What is Harry doing tomorrow?", which requests a piece of information of a future tense nature, but it is impossible to request for such a piece of information without having the initial verb of the sentence be in the relevant tense if no time related noun such as "yesterday", "tomorrow" and "later", is used. When an informational piece is being searched for, RONE often needs to pick the appropriate past tense verbs for a match in the knowledge base since when information is imparted by users it is often in a different form. For example, a user giving a piece of information would probably say, "Harry ran away.", but a user asking about that piece of information would ask, "Did Harry run away?". Notice that while the given information uses the past tense verb 'ran', the yes/no question is dominated by the initial verb 'did' and thus uses the present tense verb 'run'. In order to perform successful matches in the knowledge base, RONE replaces certain verbs to denote the proper tense.

1.4 Recognizing Verb Patterns

Assembling the participles and tenses of verb is performed based on the end patterns of the verb. Any participle or tense can be derived from any other participle or tense of the same base verb.

There are a few exceptions that are ignored which are the following:
"did", "should", "must", "could", "have", "do", "does", "would", "can", "has", "will", "are", "is", "need", "shall", "was", "were", "had", "am". These are ignored since their participles and tenses are built into the system first since they are critical in determination of sentence tense, [11].

2.0 DETERMINIG THE BASE VERB

The primary step in our process is the determination of the base verb. Doing so requires either:

1. Doing nothing if the verb is already in its base state,

2. Converting a participle to the base verb, or
3. Converting a verb tense to the base verb.

When considering the participle or tense, apart from the verb is, whose future tense is 'will' and future participle if 'will be', we observed that no other verb in the English language has a future tense or future participle. All the verbs simply attach the word "will" to their respective base verbs to become future tenses and participles. Sometimes the past participles are formed by adding the verb "was" to the base verb of the intended participle. For example the base verb "run" has a past tense "ran", a past participle of "was running", a present participle of "running", a future tense of "will run" and a future participle of "will be running". Thereby only the base verb, past tense, and present participle need to be determined in the algorithm.

2.1 Non Base Verb Markers

Converting a participle or tense to a base verb involves first determining if the word in question is in fact a not a base verb. We found that verb participles and tenses are clearly "marked" when looking at the verb backwards. All verb participles in the English Language end with "ing", or "s", and all verb past tenses end with most commonly an "ed". There are some unique cases of verb past tenses ending in "id" or "ade" and etcetera.

Table 1: Patterns of Participles and Tenses of Verbs

Key: => becomes
! not (EXCEPTION to the rule)
== equals
!= not equals

Classifications (ends with)	Sub-Classes (ends with)	Variants (ends with)	Action	Examples
S *PRESENT PARTICIPLE*	ies	== dies, ties	Remove last character	Ties => tie
		All others	replace last 3 characters with 'y'	Carries => carry
	us	none	No action	focus
	es	Vowel + consonant + es	Remove last character	Scores = score
		others	Remove last 2 characters	
It *PAST TENSE*	it	== bit	Add 'e'	Bit => bite
Ought *PAST TENSE/ PAST PARTICIPLE*	thought		Replace last 5 characters with 'ink'	thought => think
	fought		Replace last 5 characters with 'ight'	Fought => fight
	sought		Replace last 5 characters with 'eek'	Sought => seek
	bought		Replace last 5 characters with 'uy'	Bought => buy
	brought		Replace last 5 characters with	Brought =>

			'ing'	bring
Ang *PAST TENSE/ PAST PARTICIPLE*	Consonant + Sang, Consonant + rang, Consonant + tang, Consonant + wang		Replace last 3 characters with 'ing'	Sang => sing
Aught *PAST TENSE/ PAST PARTICIPLE*	caught		Replace last 3 characters with 'tch'	Caught => catch
	taught		Replace last 4 characters with 'each'	Taught => teach
Wn *PAST PARTICIPLE*	R +Vowel + wn, s +Vowel + wn, h +Vowel + wn, n +Vowel + wn, l +Vowel + wn	!=Drown, !=clo wn, !=crown, != disown, != frown	Remove last character.	Grown => grow
Ew *PAST TENSE*	Blew		Replace last 2 characters with 'ow'	Blew => blow
	flew		Replace last 2 characters with 'y'	Flew => fly
	drew		Replace last 2 characters with 'aw'	Drew => draw
Ept *PAST TENSE/ PAST PARTICIPLE*		!=accept	Replace last 3 characters with 'eep'	Kept => keep
Ting *PRESENT PARICIPLE*	Iting, ating, outing, uoting		Replace last 3 characters with 'e'	Uniting => unite
	eating	!= eating	Replace last 3 characters with 'e'	Creating => create
	others		Remove last 3 characters	Voting => vote
ning (!ening) *PRESENT PARTICIPLE*	nning		Remove last 4 characters	Running => run
	uning, oning, ining, caning		Replace last 3 characters with 'e'	Tuning => tune
	others		Remove last 3 characters	Burning => burn

Ing *PRESENT PARTICIPLE*	aking	eaking	Remove last 3 characters	Speaking => speak
		others	Replace last 3 characters with 'e'	Shaking => shake
	Vowel + consonant + ing		Replace last 3 characters with 'e'	Riding => ride
	lving, dging, gling, tling, ching, nging, bling, kling		Replace last 3 characters with 'e'	Sprinkling => sprinkle
	others		Remove last 3 characters	Hearing => hear
D *PAST TENSE/ PAST PARTICIPLE*	!dd, !rd, !ld, !nd, !vowel + d		Remove last character	Heard => hear
Ed *PAST TENSE*	Gned, yed, ned, hed	nned	Remove last 3 characters	Banned => ban
		Consonant + Vowel + ned	Remove last character	hydroplaned => hydroplane
		Vowel + Vowel + ned	Remove last 2 characters	Bemeaned => bemean
		ched	Remove last character	Psyched => psych
	Led, bed	lled	Remove last 2 characters	Swelled => swell
		bbed	Remove last 3 characters	Stubbed => stub
		Consonant + Vowel + "sub-class"	Remove last character	Prescribed => prescribe
		Vowel + Vowel + "sub-class"	Remove last 2 characters	Pooled => pool
	Cked, rked, ssed		Remove last 2 characters	Passed => pass
	rred	Vowel + rred	Remove last 3 characters	Inferred => infer
	Med	Vowel + med	Remove last character	Timed => time

		mmed	Remove last 3 characters	Crammed => cram
	ured		Remove last character	Cured => cure
	ied	!died	Replace last 3 characters with ‘y’	Unified => unify
	red	Ared, ered, ired, ored	Remove last character	Stored => store
		Uired, tred	Remove last character	Acquired => acquire
	Tted, dded	!added	Remove last 3 characters	Batted => bat
	Vowel + ted	Oated, ooted, eeted, ieted, eited	Remove last 2 characters	Footed => foot
		dited	Remove last 2 characters	Edited => edit
		others	Remove last character	Violated => violate
	Ded, ved, ged, sed, ked, zed, wed, !=wed	Vowel + gged	Remove last 3 characters	Drugged => drug
		lked	Remove last 2 characters	Talked => talk
		Vowel + wed	Remove last 2 characters	Gnawed => gnaw
		others	Remove last character	Smoked => smoke
Id *PAST PARTICIPLE*	Aid, !=aid		Replace last 2 characters with ‘y’	Laid => lay
De *PAST TENSE/ PAST PARTICIPLE*	made		Replace last 2 characters with ‘ke’	Made => make
	bade		Replace last 3 characters with ‘id’	Bade => bid

3.0 ALGORITHM USAGE

The patterns seen in Table 1 were implemented in a Java application for testing. The input was checked for the end patterns using “if else” conditions. A list of some 100 000 verbs, [18], was run through using the patterns shown in Table 1.

Usage of the patterns is based on the following algorithm, represented in first-order logic, where an input verb (represented by *a*) is compared to the Classification (represented by *x*) and to the Sub-classes (represented by *y*) and to the variant (represented by *z*).

We make the following assumptions. First, that there is at least one verb, *a*, in the English language where pattern *x* occurs.

$\exists a\ (verb(a) \rightarrow patternOccurs(a,x))$ (1)

Second, that there is at least one verb, *a*, in the English language where both pattern *x* and pattern *y* occur.

$\exists a\ (verb(a) \rightarrow patternOccurs\ (a,x) \wedge patternOccurs\ (a,y))$ (2)

Third, that there is at least one verb, *a*, in the English language where pattern *x* and pattern *y* and pattern *z* occur.

$\exists a\ (verb(a) \rightarrow patternOccurs\ (a,x) \wedge patternOccurs\ (a,y) \wedge patternOccurs\ (a,z))$

(3)

Fourth, for all *z*, if the *z* is not specified (blank entry in Table 1) and no other corresponding *z* matched, then *z* is considered to occur in *a*.

$\exists z_{1,2,3\ldots n}\ ((\neg patternOccurs\ (a,z_{1,2,3\ldots n-1}) \wedge \neg specified\ (z_n)) \rightarrow patternOccurs\ (a,z))$

(4)

Fifth, that for all *a* if pattern *x* and pattern *y* and pattern *z* occur, then *e* will not occur.

$\exists a\ ((\ patternOccurs\ (a,x) \wedge patternOccurs\ (a,y) \wedge patternOccurs\ (a,z)) \leftrightarrow$

$\neg patternOccurs(a,e))$ (5)

Therefore, if *a* falls into a pattern (*x*, *y*, *z*), then the corresponding action (represented by *b*) is taken if and only if any EXCEPTION to the rule (represented by *e*) does not occur.

changeVerb (a, x, y, z, b, e) =

patternOccurs (a,x) $\wedge$ *patternOccurs (a,y)* $\wedge$ *(patternOccurs (a,z)* $\wedge$

(¬patternOccurs (a,z) $\wedge \neg$*specified (z)))* $\wedge \neg$*patternOccurs(a,e)*

Table 2 shows a random sampling of the complete results of the simulations. Based on, [18], the results are entirely correct.

Table 2: Random Test on Base Verb Generating Algorithm

Random Test 1 (Test tense/participle => generated Base Verb)	Random Test 2 (Test tense/participle => generated Base Verb)
allying => ally	accusing => accuse
anchylosed => anchylose	aromatizing => aromatize
averaged => average	autotomising => autotomise
backsplicing => backsplice	brabbled => brabble
brutalizing => brutalize	canoed => canoe
carnifying => carnify	caravanning => caravan
ceased => cease	cold-chiselling => cold-chisell
chroming => chrome	curing => cure
confiscated => confiscate	dabbled => dabble
crapping => crap	deoxidised => deoxidise
denunciated => denunciate	diphthongizing => diphthongize
ensured => ensure	disprizing => disprize
evolving => evolve	divinized => divinize
gnawn => gnaw	elegized => elegize
halogenated => halogenate	encapsulating => encapsulate
hoeing => hoe	enthroned => enthrone
installing => install	flared => flare
jargonizing => jargonize	frivolled => frivoll
jogging => jog	ideating => ideate
meditating => meditate	illiberalizing => illiberalize
overcomplicated => overcomplicate	inosculated => inosculate
pastoralizing => pastoralize	marshalled => marshall
photoengraved => photoengrave	outplodding => outplod
preimitated => preimitate	outvoicing => outvoice
redisputed => redispute	overcultivated => overcultivate
relosing => relose	overidentified => overidentify
remortgaging => remortgage	preadvertised => preadvertise
retraversing => retraverse	prepledged => prepledge
tenderizing => tenderize	prequarantining => prequarantine
terminated => terminate	prerefining => prerefine
underpopulating => underpopulate	quasi-admiring => quasi-admire
upswept => upsweep	quoting => quote
vinylated => vinylate	recompensed => recompense

warbling => warble	reinduced => reinduce reutilized => reutilize sanitized => sanitize skywrote => skywrite surcharged => surcharge unfenced => unfence upttore => upttear vocalized => vocalize zigzagged => zigzag

4.0 DISCUSSION AND CONCLUSION

Although the tests so far have indicated that any verb can be decomposed correctly into its base state, should any verb be decomposed incorrectly, the correction can be performed by adding another condition to Table 1. An advantage of being able to obtain the base verb is the possibility that synonyms and antonyms of these base verbs are more readily accessible. The usefulness in obtaining the base verb also lies in that once the base verb has been obtained, the participles and tenses can be expanded using a similar technique of end patterns. We are currently working on such an algorithm and will publish our results as soon as we have conclusive results.

REFERENCES

[1] R.G. Raj. "Using SQL databases as a viable memory system for a learning AI entity", *in proceedings of the 3rd International Conference on Postgraduate Education (ICPE-3'2008)*, Penang, Malaysia, (Paper ID: BRC 160), 2008.

[2] P. Merlo and S. Stevenson. "Automatic verb classification based on statistical distributions of argument structure". *Computational Linguistics*, 2001, Vol. 27(3), pp 373–408.

[3] A. Korhonen, Y. Krymolowski and N. Collier. "Automatic classification of verbs in biomedical texts", *in Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual meeting of the ACL*, 2006b, pp 345–352.

[4] S. Schulte im Walde. "Experiments on the automatic induction of german semantic verb classes". *Computational Linguistics*, 2006, Vol. 32(2), pp 159–194.

[5] B. J. Dorr. "Large-scale dictionary construction for foreign language tutoring and interlingual machine translation". *Machine Translation*, 1997, Vol. 12(4), pp 271–322.

[6] D. Prescher, S. Riezler and M. Rooth. "Using a probabilistic class-based lexicon for lexical ambiguity resolution", *in proceedings of the 18th International Conference on Computational Linguistics*, Saarbr¨ucken, Germany, 2000, pp 649–655.

[7] R. Swier and S. Stevenson. "Unsupervised semantic role labeling", *in Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102, Barcelona, Spain, August 2004.

[8] H. T. Dang. "*Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*". Ph.D. thesis, CIS, University of Pennsylvania, 2004.

[9] L. Shi and R. Mihalcea (2005). "Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing", *in Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, Mexico, 2005.

[10] R. Jackendoff. "*Semantic Structures*". MIT Press, Cambridge, Massachusetts, 1990.

[11] B. Levin. "*English Verb Classes and Alternations*". Chicago University Press, Chicago, 1993.

[12] G. A. Miller. "WordNet: An on-line lexical database". *International Journal of Lexicography*, Vol. 3(4), 1990, pp 235–312.

[13] S. Schulte im Walde. "Clustering verbs semantically according to their alternation behaviour", *in Proceedings of COLING*, Saarbr¨ucken, Germany, 2000, pp 747–753.

[14] A. Korhonen, Y. Krymolowski and T. Briscoe. "A large subcategorization lexicon for natural language processing applications", *in Proceedings of LREC*, 2006a.

[15] P. Nakov and M. Hearst. "*Using Verbs to Characterize Noun-Noun Relations*". [online] available at: **http://biotext.berkeley.edu**/papers/aimsa2006.pdf, 2006.

[16] I.S. Bajwa, A. Samad and S. Mumtaz. "Object Oriented Software Modeling Using NLP Based Knowledge Extraction," in *European Journal of Scientific Research*, ISSN 1450-216X, 2009, Vol.35 No.1, pp 22-33.

[17] R.G. Raj and S. Abdul-Kareem, "Information Dissemination And Storage For Tele-Text Based Conversational Systems' Learning". *Malaysian Journal of Computer Science*, December 2009, Vol. 22(2): pp 138-159.

[18] G. Ward. "*mobypos.txt, Moby Word Lists*", Project Gutenberg Literary Archive Foundation, [online] available at: http://www.gutenberg.org/etext/3201, 2002.