FROM TEXT TO SPEECH
THE MITALK SYSTEM

Presented by
Najeeb Khan
2013-5-24

MORPHOLOGICAL ANALYSIS ALGORITHM

MORPHOLOGICAL ANALYSIS ALGORITHM

- The decomposition algorithm consists of three major components
  - A recursive morph partitioning algorithm
  - A set of spelling change rules for use at morph boundaries
  - A set of selectional rules to distinguish between legal and illegal morph sequences and to choose the best morph covering when multiple coverings exist

Recursive Morph Decomposition

RECURSIVE MORPH DECOMPOSITION

- The right end of each word is matched against the longest lexicon morph possible

RECURSIVE MORPH DECOMPOSITION

- The right end of each word is matched against the longest lexicon morph possible
- The procedure is recursively invoked on the remaining uncovered portion of the word

RECURSIVE MORPH DECOMPOSITION

- The right end of each word is matched against the longest lexicon morph possible
- The procedure is recursively invoked on the remaining uncovered portion of the word
- If this recursive invocation fails to produce covering, then the original match is discarded and the next longest matching morph is used

RECURSIVE MORPH DECOMPOSITION

RECURSIVE MORPH DECOMPOSITION

- Input to the decomposition procedure consists of
  - A word or remainder to be covered
  - A state flag that indicates which morph types are legal in the current context
  - A score value that is used to rank multiple decompositions

RECURSIVE MORPH DECOMPOSITION

find the longest morph which matches the right end of the current string

WHILE there is a match DO

IF the matching morph is compatible with the current context (state)

THEN remove the matched letters from the right side of the string,

update the current state and score as a function of the type of the matched morph.

find a set of possible spelling changes[1] at the right end of the remainder,

attempt a recursive decomposition for each spelling variation,

save the results of the best-scoring of these variations,

restore the remainder string, state, and score to their original values.

END IF,

find the next longest morph which matches the right end of the string.

END WHILE.

MORPH TYPES

MORPH TYPES

- Not all sequences of morphs are legal in the English language

Morph Types

- Not all sequences of morphs are legal in the English language
- For this reason each morph in the lexicon has a type code

MORPH TYPES

- Not all sequences of morphs are legal in the English language
- For this reason each morph in the lexicon has a type code
- FREE ROOT: a word which can appear alone or with suffixes, prefixes and/or other roots e.g. side, cover, spell

MORPH TYPES

- Not all sequences of morphs are legal in the English language
- For this reason each morph in the lexicon has a type code
- FREE ROOT: a word which can appear alone or with suffixes, prefixes and/or other roots e.g. side, cover, spell
- ABSOLUTE: words which do not allow most suffixes e.g. the, into, of, proper names

MORPH TYPES

- Not all sequences of morphs are legal in the English language
- For this reason each morph in the lexicon has a type code
- FREE ROOT: a word which can appear alone or with suffixes, prefixes and/or other roots e.g. side, cover, spell
- ABSOLUTE: words which do not allow most suffixes e.g. the, into, of, proper names
- PREFIX: a prefix which can combine with roots and other prefixes e.g. pre, dis, mis

MORPH TYPES

MORPH TYPES

- Suffixes are identified using two different criteria

MORPH TYPES

- Suffixes are identified using two different criteria
- DERIVATIONAL: these have a major effect on the meaning of the root and may change the part of speech e.g. ness, ment, y

MORPH TYPES

- Suffixes are identified using two different criteria
- DERIVATIONAL: these have a major effect on the meaning of the root and may change the part of speech e.g. ness, ment, y
- INFLECTIONAL: these suffixes change only the tense, number or inflection of the root e.g. ing, ed, s

MORPH TYPES

- Suffixes are identified using two different criteria
- DERIVATIONAL: these have a major effect on the meaning of the root and may change the part of speech e.g. ness, ment, y
- INFLECTIONAL: these suffixes change only the tense, number or inflection of the root e.g. ing, ed, s
- VOCALIC/NONVOCALIC: depends on whether the suffix begins with a vowel or consonant

MORPH TYPES

MORPH TYPES

- STRONG: a root a which already containd tense or number information e.g. went(go+past) and women(woman+plural)

MORPH TYPES

- STRONG: a root a which already containd tense or number information e.g. went(go+past) and women(woman+plural)
- LF-ROOT: a root which must be followed by a derivational suffix e.g. absorpt in absorptive and absorption

MORPH TYPES

- STRONG: a root a which already containd tense or number information e.g. went(go+past) and women(woman+plural)
- LF-ROOT: a root which must be followed by a derivational suffix e.g. absorpt in absorptive and absorption
- RF-ROOT: a root which must be preceeded by a prefix e.g. mit in permit, submit, transmit

Legal Morph Sequences

LEGAL MORPH SEQUENCES

- The detection of legal and illegal morph sequences is performed by a finite state machine (FSM)

LEGAL MORPH SEQUENCES

- The detection of legal and illegal morph sequences is performed by a finite state machine (FSM)
- The grammar recognized by the FSM is summarized in production rules given

effective-root = ROOT | LF-ROOT DERIV | PREFIX RF-ROOT | STRONG

suffix = DERIV | INFL

affixed-word = { PREFIX } effective-root { suffix }

absolute-word = ABSOLUTE | ABSOLUTE INFL { suffix } | INITIAL affixed-word

word = affixed-word | absolute-word

compound-absolute = absolute-word | absolute-word HYPHEN compound | ABSOLUTE INFL { suffix } compound-affixed

compound-affixed = affixed-word | affixed-word HYPHEN compound | affixed-word compound-affixed

compound = compound-affixed | compound-absolute

Legal Morph Sequences

LEGAL MORPH SEQUENCES

- Each state of the FSM represents a summary of the type of the morphs which have been stripped from the word being decomposed

LEGAL MORPH SEQUENCES

- Each state of the FSM represents a summary of the type of the morphs which have been stripped from the word being decomposed
- It is this state which is passed as a parameter to the recursive decomposition procedure

SELECTIONAL RULES AND SCORING

SELECTIONAL RULES AND SCORING

- When multiple morph coverings are found, selectional rules are needed to choose the covering most likely to be correct

SELECTIONAL RULES AND SCORING

- When multiple morph coverings are found, selectional rules are needed to choose the covering most likely to be correct
- The standard form for a possibly compound word is stated as two production rules

SELECTIONAL RULES AND SCORING

- When multiple morph coverings are found, selectional rules are needed to choose the covering most likely to be correct
- The standard form for a possibly compound word is stated as two production rules
- Std-root= (ROOT|LF-ROOT DERIV)

SELECTIONAL RULES AND SCORING

- When multiple morph coverings are found, selectional rules are needed to choose the covering most likely to be correct
- The standard form for a possibly compound word is stated as two production rules
- Std-root= (ROOT|LF-ROOT DERIV)
- Std-root= {PREFIX}{std-root}
(std-root {ROOT}|STRONG) {INFL}

SELECTIONAL RULES AND SCORING

- When multiple morph coverings are found, selectional rules are needed to choose the covering most likely to be correct
- The standard form for a possibly compound word is stated as two production rules
- Std-root= (ROOT|LF-ROOT DERIV)
- Std-root= {PREFIX}{std-root}
  (std-root {ROOT}|STRONG) {INFL}
- Coverings which match this form are to be preferred above all others

SELECTIONAL RULES AND SCORING

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found
- PREFIX + ROOT > ROOT + DERIV >ROOT + INFL >ROOT + ROOT

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found
- PREFIX + ROOT > ROOT + DERIV >ROOT + INFL >ROOT + ROOT
- PREFIX + PREFIX + ROOT > ROOT + ROOT

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found
- PREFIX + ROOT > ROOT + DERIV >ROOT + INFL >ROOT + ROOT
- PREFIX + PREFIX + ROOT > ROOT + ROOT
- ROOT + DERIV + DERIV > ROOT + ROOT

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found
- PREFIX + ROOT > ROOT + DERIV >ROOT + INFL >ROOT + ROOT
- PREFIX + PREFIX + ROOT > ROOT + ROOT
- ROOT + DERIV + DERIV > ROOT + ROOT
- These rules are implemented by associating a cost with each transition of FSM and keeping track of the total cost of decomposition

SELECTIONAL RULES AND SCORING

- Among coverings that match the standard form the following partial ordering were found
- PREFIX + ROOT > ROOT + DERIV >ROOT + INFL >ROOT + ROOT
- PREFIX + PREFIX + ROOT > ROOT + ROOT
- ROOT + DERIV + DERIV > ROOT + ROOT
- These rules are implemented by associating a cost with each transition of FSM and keeping track of the total cost of decomposition
- The covering with lowest cost is the most desirable

RECOGNIZING MORPHOLOGICAL MUTATION

RECOGNIZING MORPHOLOGICAL MUTATION

- After a suffix has been removed from a word it is necessary to investigate possible spelling changes which may have taken place during composition e.g.
  - Y→I (embody-ment → embodiment)
  - Consonant doubling before a vocalic suffix (pad-ing → padding)
  - Dropping silent e before a vocalic suffix (fire-ing → firing)

RECOGNIZING MORPHOLOGICAL MUTATION

RECOGNIZING MORPHOLOGICAL MUTATION

- Different morphs have different behavior in the presence of change-causing suffixes

RECOGNIZING MORPHOLOGICAL MUTATION

- Different morphs have different behavior in the presence of change-causing suffixes
- Three categories of morph behavior are defined
  - Required: scar+ed → scarred
  - Forbidden: alloy+ing → alloying
  - Optional: change+able → changeable,
    change+ing → changing

RECOGNIZING MORPHOLOGICAL MUTATION

Recognizing Morphological Mutation

- The spelling changes performed by DECOMP consists of appending or deleting the last letter

RECOGNIZING MORPHOLOGICAL MUTATION

- The spelling changes performed by DECOMP consists of appending or deleting the last letter
- If the matched morph is a vocalic suffix then spelling changes are performed by a three character template against the last two remainder letters and the first letter of the matched morph.

Pattern	Change	Example
ck+.	none	packing → pack+ing
	ck → c	picnicking → picnic+ing
xx+i	none	telling → tell+ing
	xx → x	padding → pad+ing
	xx → xxe	silhouetting → silhouette+ing
xx+.	none	yeller → yell+er
	xx → x	reddest → red+est
e+e	e → ee (+)	freed → free+ed
e+i	none	dyeing → dye+ing
e+.	none	changeable → change+able
i+i	none	skiing → ski+ing
i+e	i → y	noisiest → noisy+est
	i → ie (+)	eeriest → eerie+est
	none	efficient → effici+ent
i+.	i → y	variation → vary+ation
	none	deviate → devi+ate
y+i	none (+)	flying → fly+ing
	y → ye (+)	eying → eye+ing
y+.	none	employer → employ+er
.+i	→ e	daring → dare+ing
	none	showing → show+ing
	→ y (*)	harmonize → harmony+ize
.+.	→ e	observance → observe+ance
	none	sender → send+er

RECOGNIZING MORPHOLOGICAL MUTATION

RECOGNIZING MORPHOLOGICAL MUTATION

- For each possible spelling of the remainder the following steps are performed
  - Make the change
  - Recursively decompose the remainder
  - If a morph matches the right end of the remainder, check its spelling change code to see if it is compatible with the change

```
Decomp: "SCARCITY" [state = word <0> inflectional suffix] =>
Decomp:  Matched "CITY" (root) -- decompose remainder
Decomp:   "SCAR" [state = <101> root] =>
Decomp:    Matched "SCAR" (root) -- decompose remainder
Decomp:     "" [state = <234> root] =>
Decomp:      Matched start of word, final score = 234
Decomp:    Matched "CAR" (root) min. score = 268 -- too expen-
     sive!
Decomp:    Matched "AR" (derivational suffix) min. score = 234
     -- too expensive!
Decomp:  Matched "ITY" (derivational suffix) -- decompose
     remainder
Decomp:   "SCARCE" [state = root <35> derivational suffix] =>
Decomp:    Matched "SCARCE" (root) -- decompose remainder
Decomp:     "" [state = <136> root] =>
Decomp:      Matched start of word, final score = 136
Decomp:   "SCARC" [state = root <35> derivational suffix] =>
Decomp:    Matched "ARC" (root) min. score = 170 -- too expen-
     sive!
Decomp:   "SCARCY" [state = root <35> derivational suffix] =>
Decomp:    Matched "Y" (derivational suffix) min. score = 136 --
     too expensive!
Decomp:  Matched "Y" (derivational suffix) -- decompose
     remainder
Decomp:   "SCARCITE" [state = root <35> derivational suffix] =>
Decomp:    Matched "CITE" (root) min. score = 170 -- too expen-
     sive!
Decomp:   "SCARCIT" [state = root <35> derivational suffix] =>
Decomp:    Matched "IT" (absolute) -- illegal!
DECOMP: SCARCITY                                  word spelling
DECOMP:     NOUN (NUMBER = SINGULAR)              part of speech and features
DECOMP:   =>                                      decomposition follows
DECOMP:     SCARCE [ROOT] :                       first morph spelling and type
DECOMP:         1SKE*RS (ADJECTIVE)               pronunciation and part of speech
DECOMP:     ITY [DERIVATIONAL VOCALIC SUFFIX] :   second morph
DECOMP:         *T-E^ (NOUN)
```

PHRASE LEVEL PARSER

PHRASE LEVEL PARSER

- The TTS parser supplies a surface structure parse, providing information for algorithms which produce prosodic effects in the output speech

PHRASE LEVEL PARSER

- The TTS parser supplies a surface structure parse, providing information for algorithms which produce prosodic effects in the output speech
- Phrase recognition is accomplished via an Augmented Transition Network ATN and the grammars for noun groups and verb group

PHRASE LEVEL PARSER

PHRASE LEVEL PARSER

- NGR: pronoun(him, several), pronoun with modification(almost anything green), an integer (nearly hundred thousand), a noun phrase up to head noun(his own red and black car) or any of the above preceded by a preposition

PHRASE LEVEL PARSER

- NGR: pronoun(him, several), pronoun with modification(almost anything green), an integer (nearly hundred thousand), a noun phrase up to head noun(his own red and black car) or any of the above preceded by a preposition
- VGR: verb phrase without direct or indirect objects (could almost see, had been very yellow)

PHRASE LEVEL PARSER

- NGR: pronoun(him, several), pronoun with modification(almost anything green), an integer (nearly hundred thousand), a noun phrase up to head noun(his own red and black car) or any of the above preceded by a preposition
- VGR: verb phrase without direct or indirect objects (could almost see, had been very yellow)
- VBL: verbal group which is either infinitive phrase (to walk slowly) or a participle phrase

PHRASE LEVEL PARSER

PHRASE LEVEL PARSER

- Input

PHRASE LEVEL PARSER

- Input
  - The input to the parser contains the morph spelling, morph pronunciation, morph type and parts of speech and features for each homograph of each morph in the analysis of the word

PHRASE LEVEL PARSER

- Input
  - The input to the parser contains the morph spelling, morph pronunciation, morph type and parts of speech and features for each homograph of each morph in the analysis of the word
- Output
  - For each node, the number of words covered by that node, the pos of the node, and a property list is given
  - Each word is accompanied by its spelling and pos set
  - For each pos a property list is given

PARTS OF SPEECH

PARTS OF SPEECH

- The following are the parts of speech of the open class words

VERB (INF TR) (PL TR) = infinitive form of verb

VERB (SING TR) (PL TR) = past tense verb

ADJ = adjective

ADV (VMOD TR) (ADJMOD TR) =
adverb which can modify either an adjective or a verb

ADV (ADJMOD TR) = adverb which can modify an adjective

PREP = preposition

CONJ = conjunction

INTG = integer

INTG (NUM SING) = one

INTG (DEF FL) = integer which requires a (e.g. thousand)

VERBING = present participle

VERBEN = past participle

TO = to

SCONJ = sentential conjunction (e.g. whether)

CONTR = contraction (e.g. 're)

INTERJ = interjection (e.g. oh)

PARTS OF SPEECH PROCESSOR

PARTS OF SPEECH PROCESSOR

- The POS processor computes POS for each word in the input given the morph decomposition and the pos of the mmorphs

PARTS OF SPEECH PROCESSOR

- The POS processor computes POS for each word in the input given the morph decomposition and the pos of the mmorphs
- It is based on the Allen’s Preprocessor

PARTS OF SPEECH PROCESSOR

PARTS OF SPEECH PROCESSOR

- The current algorithm goes right to left across morphs and uses the part of speech of the rightmost morph for a compound as well as for the cases where there is a suffix

PARTS OF SPEECH PROCESSOR

- The current algorithm goes right to left across morphs and uses the part of speech of the rightmost morph for a compound as well as for the cases where there is a suffix
- This is justified by the facts that
  - Suffixes determine the pos of a word with regularity e.g. …ness is a NOUN
  - The POS of compounds is very idiosyncratic and the best heuristic is to use the POS set of the rightmost root

THE PARSER ALGORITHM

THE PARSER ALGORITHM

- The operation of the parsing logic can be thought of as having two levels

THE PARSER ALGORITHM

- The operation of the parsing logic can be thought of as having two levels
- The global level reflects the parsing strategy, which has been found to give the best phrases, it is base on three empirical facts
  - There are more noun groups than verb groups
  - The initial portions of noun groups are easier to detect than verb groups. Verb groups frequently begins with the verb itself which often has both NOUN and VERB in its POS set

THE PARSER ALGORITHM

- The operation of the parsing logic can be thought of as having two levels
- The global level reflects the parsing strategy, which has been found to give the best phrases, it is base on three empirical facts
  - There are more noun groups than verb groups
  - The initial portions of noun groups are easier to detect than verb groups. Verb groups frequently begins with the verb itself which often has both NOUN and VERB in its POS set
- The local level merely interprets the ATN grammar

THE PARSER ALGORITHM

THE PARSER ALGORITHM

- The global parsing strategy proceeds as follows

THE PARSER ALGORITHM

- The global parsing strategy proceeds as follows
- It looks for the longest noun group that it can find beginning with the first word in the sentence

THE PARSER ALGORITHM

- The global parsing strategy proceeds as follows
- It looks for the longest noun group that it can find beginning with the first word in the sentence
- If it locates one then a node is constructed and the current word pointer is incremented

THE PARSER ALGORITHM

- The global parsing strategy proceeds as follows
- It looks for the longest noun group that it can find beginning with the first word in the sentence
- If it locates one then a node is constructed and the current word pointer is incremented
- If it fails it attempts to find the longest group starting at the word pointer and creates a node if it finds a verb phrase

THE PARSER ALGORITHM

- The global parsing strategy proceeds as follows
- It looks for the longest noun group that it can find beginning with the first word in the sentence
- If it locates one then a node is constructed and the current word pointer is incremented
- If it fails it attempts to find the longest group starting at the word pointer and creates a node if it finds a verb phrase
- If neither type of group is found the word pointer is simply incremented and the process begins again

The Parser Algorithm

THE PARSER ALGORITHM

- At the local level, the parser uses the ATN to find a constituent

THE PARSER ALGORITHM

- At the local level, the parser uses the ATN to find a constituent
- There are two pointers, one pointing to the word in the sentence currently being examined and one pointing to the current state in the net

THE PARSER ALGORITHM

- At the local level, the parser uses the ATN to find a constituent
- There are two pointers, one pointing to the word in the sentence currently being examined and one pointing to the current state in the net
- The parser tries each arc leading from the current state in the order in which they appear in the net

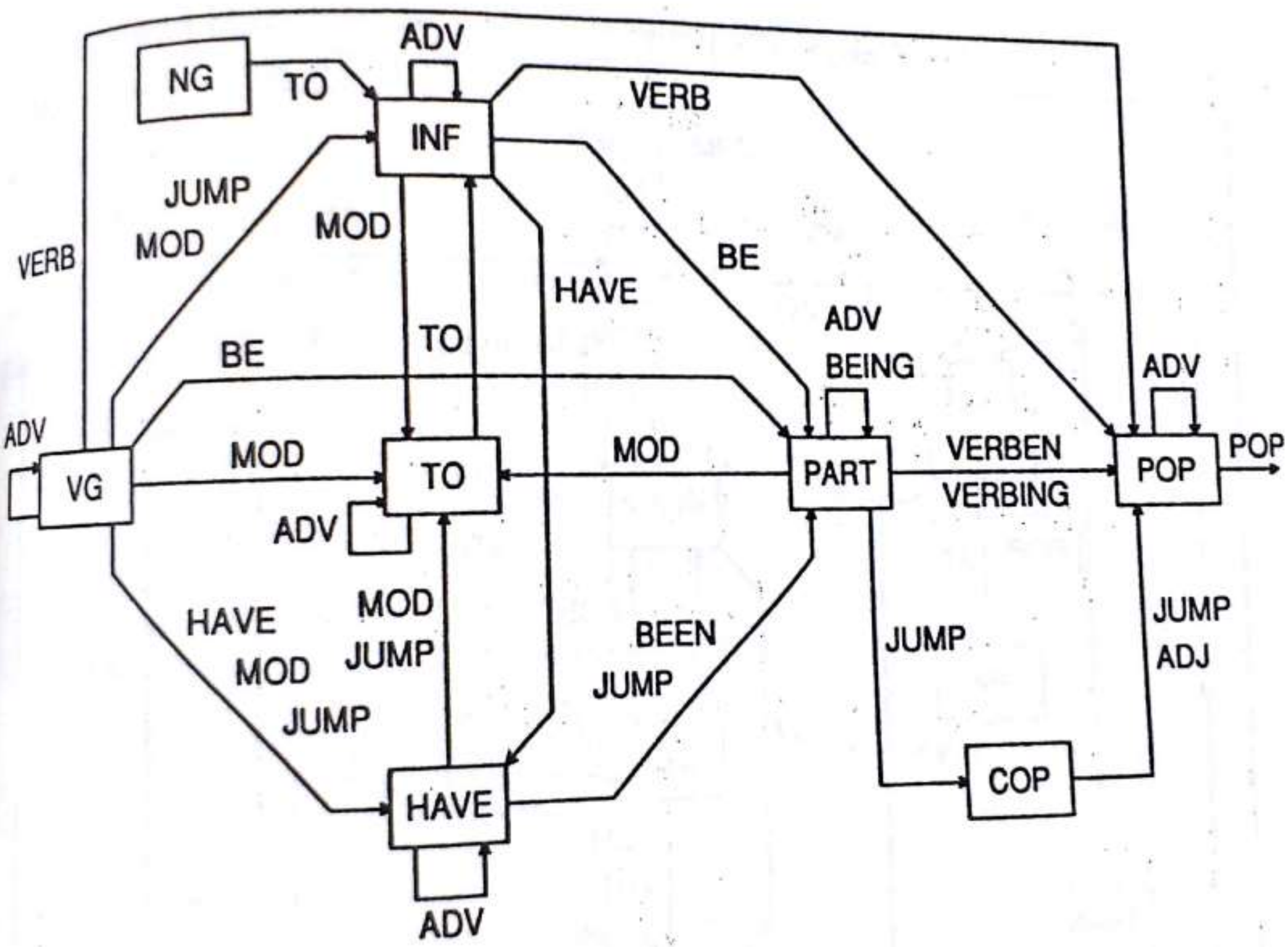
THE PARSER ALGORITHM

THE PARSER ALGORITHM

- Testing an arc is done as follows
  - If the arc is JUMP or POP, then the exit routine associated with that arc is tested. If it is successful then
  - For JUMP the state pointer is advanced to destination
  - For POP a node is built if the POP constituent is longer than any found so far

THE PARSER ALGORITHM

- Testing an arc is done as follows
  - If the arc is JUMP or POP, then the exit routine associated with that arc is tested. If it is successful then
  - For JUMP the state pointer is advanced to destination
  - For POP a node is built if the POP constituent is longer than any found so far
- If the arc is a part of speech and the current word does not have this pos, the parser continues with the next arc


NG
TO
ADV
INF
VERB
JUMP
MOD
MOD
VERB
HAVE
BE
ADV
BEING
TO
BE
ADV
ADV
MOD
MOD
TO
PART
VERBEN
VERBING
POP
POP
VG
ADV
HAVE
MOD
MOD
JUMP
BEEN
JUMP
JUMP
JUMP
ADJ
JUMP
COP
HAVE
ADV

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

- There are sometimes changes in pronunciation at morph boundaries

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

- There are sometimes changes in pronunciation at morph boundaries
- Module SOUND1 checks for contexts in which such changes occur and changes the pronunciation

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

- Input
  - Input to SOUND1 is the output of the parser. It contains the morph pronunciation information from DECOMP and phrase and pos information from PARSER

MORPHOPHONEMICS AND STRESS ADJUSTMENTS

- Input
  - Input to SOUND1 is the output of the parser. It contains the morph pronunciation information from DECOMP and phrase and pos information from PARSER
- Output
  - The output stream from SOUND1 consists of a string of phonetic segment labels, stress marks and syllable and morph boundaries for each word

Morphophonemic Rules

MORPHOPHONEMIC RULES

- The pronunciation for each word which is segmented by DECOMP is constructed by catenating the pronunciations of its component morphs

MORPHOPHONEMIC RULES

- The pronunciation for each word which is segmented by DECOMP is constructed by catenating the pronunciations of its component morphs
- The following rules are applied to modify morph pronunciations when necessary

MORPHOPHONEMIC RULES

- The pronunciation for each word which is segmented by DECOMP is constructed by catenating the pronunciations of its component morphs
- The following rules are applied to modify morph pronunciations when necessary
- Words which end in ss,zz,sh,zh,ch and jj form their plural and possessives by the concatenation of the segment string IH ZZ or in its vowel reduced form IX ZZ (busses, churches, garages)

Morphophonemic Rules

MORPHOPHONEMIC RULES

- After the voiced segments, the plural and possessive morphemes are realized as zz (dogs, potatos)

MORPHOPHONEMIC RULES

- After the voiced segments, the plural and possessive morphemes are realized as zz (dogs, potatos)
- After other unvoiced consonants it is pronounced SS (backs, cat’s)

MORPHOPHONEMIC RULES

- After the voiced segments, the plural and possessive morphemes are realized as zz (dogs, potatos)
- After other unvoiced consonants it is pronounced SS (backs, cat's)
- For past tense forms after the segments TT and DD the extra vowel separation is provided to give the pronunciation IH DD or IX DD (minted, mended)

MORPHOPHONEMIC RULES

- After the voiced segments, the plural and possessive morphemes are realized as zz (dogs, potatos)
- After other unvoiced consonants it is pronounced SS (backs, cat’s)
- For past tense forms after the segments TT and DD the extra vowel separation is provided to give the pronunciation IH DD or IX DD (minted, mended)
- After other voiced DD is chosen(whispered)

MORPHOPHONEMIC RULES

- After the voiced segments, the plural and possessive morphemes are realized as zz (dogs, potatos)
- After other unvoiced consonants it is pronounced SS (backs, cat's)
- For past tense forms after the segments TT and DD the extra vowel separation is provided to give the pronunciation IH DD or IX DD (minted, mended)
- After other voiced DD is chosen(whispered)
- After other unvoiced TT is chosen (hushed)

MORPHOPHONEMIC RULES

MORPHOPHONEMIC RULES

- Preceding the suffix ion the letter t is pronounced ch after n and s and SH otherwise (retention, congestion, completion)

MORPHOPHONEMIC RULES

- Preceding the suffix ion the letter t is pronounced ch after n and s and SH otherwise (retention, congestion, completion)
- Preceing e, i, and y the suffix ic is changed from IH KK to IH SS (electricity

STRESS MODIFICATION RULES

STRESS MODIFICATION RULES

- A compound stress rule is applied to words decomposed into more than one root

STRESS MODIFICATION RULES

- A compound stress rule is applied to words decomposed into more than one root
- The primary stress is retained on the leftmost root

STRESS MODIFICATION RULES

- A compound stress rule is applied to words decomposed into more than one root
- The primary stress is retained on the leftmost root
- Primary stress on all other nodes is reduced to secondary stress

STRESS MODIFICATION RULES

- A compound stress rule is applied to words decomposed into more than one root
- The primary stress is retained on the leftmost root
- Primary stress on all other nodes is reduced to secondary stress
- Suffixes which shift the primary stress in a word such as ee, eer, esce and ation are entered in the lexicon with primary stress

STRESS MODIFICATION RULES

- A compound stress rule is applied to words decomposed into more than one root
- The primary stress is retained on the leftmost root
- Primary stress on all other nodes is reduced to secondary stress
- Suffixes which shift the primary stress in a word such as ee, eer, esce and ation are entered in the lexicon with primary stress
- The stress on any root to which they attach is reduced to secondary (trainee, auctioneer)

LETTER TO SOUND AND LEXICAL STRESS

LETTER TO SOUND AND LEXICAL STRESS

- It is necessary to have a scheme which stipulates a pronunciation for words not analyzable by the lexical analysis algorithm

LETTER TO SOUND AND LEXICAL STRESS

- It is necessary to have a scheme which stipulates a pronunciation for words not analyzable by the lexical analysis algorithm
- This comprehensiveness is provided by the letter to sound section of SOUND1

LETTER TO SOUND AND LEXICAL STRESS

- It is necessary to have a scheme which stipulates a pronunciation for words not analyzable by the lexical analysis algorithm
- This comprehensiveness is provided by the letter to sound section of SOUND1
- The letter strings which it receives are converted into stressed phonetic segments strings using two sets of ordered phonological rules

LETTER TO SOUND AND LEXICAL STRESS

LETTER TO SOUND AND LEXICAL STRESS

- The first set to be applied converts letters to phonetic segments, first stripping affixes, then converting consonants and finally converting vowels and affixes

LETTER TO SOUND AND LEXICAL STRESS

- The first set to be applied converts letters to phonetic segments, first stripping affixes, then converting consonants and finally converting vowels and affixes
- The second set applies an ordered set of rules which determine the stress contour of the segment string

LETTER TO SOUND

- The conversion of a letter string to a phonetic segment string in the letter to sound program proceeds in three stages
- In the first stage prefixes and suffixes are detected
- Suffixes are detected right to left and prefixes are detected left to right

LETTER TO SOUND

- The conversion of a letter string to a phonetic segment string in the letter to sound program proceeds in three stages
- In the first stage prefixes and suffixes are detected
- Suffixes are detected right to left and

fin+ish+ing		*possible suffix analysis*
ing:	(a)	nominal or verbal suffix
	(b)	follows nominal or verbal suffix
ish:	(a)	adjectival suffix
		parts of speech not compatible
	(b)	follows nominal or adjectival suffix
finish+ing		*correct analysis*

LETTER TO SOUND

- Domain of application:
- The domain of application of the second stage is assumed to be a single-root morph
- This stage is intended for consonant rules and proceeds from left of the string to the right
- Extending the domain to the whole letter again for the third stage, a phonemic representation is given to affixes, vowels and vowel digraphs.
- Phonemic representations are produced by a set of ordered rules in a given context

RULE ORDERING

- Because the pronunciation of consonants is least dependent upon context, phonological rules for consonants are applied first
- Rules for vowels and affixes requiring more specification of environment are applied in the third stage
- Within the sets of rules for conversion of consonants and vowels, ordering proceeds from longer strings to shorter strings (cch then cc, ch)

APPLICATION OF LETTER TO SOUND RULES TO CARIBOU

```
Sound1: Stripped   : C  A  R  I  B  O  U      Stage 1
Sound1: Consonants: KK ?? RR ?? BB ?? ??      Stage 2
Sound1: Prefixes   : KK ?? RR ?? BB ?? ??
Sound1: Vowels     : KK AE RR IH BB UW        Stage 3
Sound1: Suffixes   : KK AE RR IH BB UW
 SOUND1:  KK 'AE RR - IX - BB UW
```

Thank You