Junyang Xin

N11176767

1. (a) Result of running against the original chunk patterns:

# of matching tags = 90; # of tags in key = 114; # of tags in response = 140;

precision = 64.28%; recall = 78.94%; type accuracy = 100.0%

(b) Result of running against the modified chunk patterns of assignment 4:

# of matching tags = 92; # of tags in key = 114; # of tags in response = 123;

precision = 74.79%; recall = 80.7%; type accuracy = 100.0%

Result of running against the further modified chunk patterns of this assignment:

# of matching tags = 94; # of tags in key = 114; # of tags in response = 111;

precision = 84.68%; recall = 82.45%; type accuracy = 100.0%

(c) Result of running against the original chunk patterns:

# of matching tags = 93; # of tags in key = 130; # of tags in response = 153;

precision = 60.78%; recall = 71.53%; type accuracy = 100.0%

Result of running against the further modified chunk patterns of this assignment:

# of matching tags = 90; # of tags in key = 130; # of tags in response = 125;

precision = 72.0%; recall = 69.23%; type accuracy = 100.0%

And the final chunk patterns are listed below:

// pattern set for noun and verb groups

pattern set chunks;

// Temporal expressoin patterns. They appear first in order to take higher

// precedence over proper noun and ngroup.

temporal-expr := clock-time | day-of-week | date-of-month | month;

clock-time := [constit cat=q] ":" [constit cat=q] "pm" |

[constit cat=q] ":" [constit cat=q] "am";

date-of-month := month [constit cat=q] | [constit cat=q] "of" month;

month := "January" | "Jan" | "February" | "Feb" | "March" | "Mar" |

"April" | "Apr" | "May" | "June" | "Jun" | "July" | "Jul" | "August" |

"Aug" | "September" | "Sep" | "October" | "Oct" | "November" | "Nov" |

"December" | "Dec";

day-of-week := "Monday" | "Mon" | "Tuesday" | "Tue" | "Wednesday" | "Wed" |

"Thursday" | "Thu" | "Friday" | "Fri" | "Saturday" | "Sat" | "Sunday" |

"Sun";

// This doesn't work as intended yet... need to look more into this.

date-of-year := [constit cat=q] ":" [constit cat=q] ":" [constit cat=q] |

[constit cat=q] "/" [constit cat=q] "/" [constit cat=q] |

date-of-month "," [constit cat=q] | date-of-month [constit cat=q];

when temporal-expr add [constit cat=temporal-expr-group];

// Patterns for greetings.

greeting := "Hey" | "hey" | "Hello" | "hello" | "Hola" | "holo" | "Goodbye" |

"goodbye" | "Hi" | "hi";

when greeting add [constit cat=greet-group];

// Patterns for addresses.

address-phrase := [constit cat=q] proper-noun facility;

facility := "Road" | "Rd" | "Street" | "Str" | "Avenue" | "Ave" | "Lane";

when address-phrase add [constit cat=addr-group];

// patterns for noun groups

ng := noun-phrase | proper-noun-phrase | pro-phrase |

proper-noun-phrase noun-phrase |

noun-phrase proper-noun-phrase;

noun-phrase :=

det-pos? [constit cat=q]? quant-colloq? adj-phrase\* proper-noun\* [constit cat=n]+ conj-noun\*;

proper-noun-phrase :=

det-pos? [constit cat=q]? quant-colloq? adj-phrase\* proper-noun\* proper-noun+ conj-proper-noun\*;

pro-phrase :=

quant-colloq? [constit cat=pro] conj-pro\*;

// Added possession form for proper nouns. Also added adj before the nouns.

det-pos := pre-det? [constit cat=det] |

pre-det? [constit cat=det]? [constit cat=adj]\* [constit cat=n number=singular] "'s" |

pre-det? [constit cat=det]? [constit cat=adj]\* proper-noun "'s";

pre-det := "such" | "what" | "rather" | "quite";

proper-noun := ([token case=cap] | [undefinedCap])+;

// Quantifier of colloquial form.

quant-colloq := [constit cat=n] "of" | [constit cat=q] "of" | proper-noun "of";

conj-noun := "and" [constit cat=n]+;

conj-proper-noun := "and" proper-noun+;

conj-pro := "and" [constit cat=pro];

adj-phrase := [constit cat=adj] conj-adj\*;

conj-adj := [constit cat=cconj] [constit cat=adj];

when ng add [ngroup];

// patterns for active verb groups

vg := [constit cat=tv] |

[constit cat=w] vg-inf |

[constit cat=w] "do" |

tv-vbe vg-ving;

vg-inf := [constit cat=v] |

"be" vg-ving;

vg-ven := [constit cat=ven] |

"been" vg-ving |

"been" [constit cat=ven];

vg-ving := [constit cat=ving] |

"being" [constit cat=ven];

tv-vbe := "is" | "are" | "was" | "were";

tv-vbp := "have" | "has" | "had";

when vg add [constit cat=vgroup];

// patterns for passive verb groups

vg-pass := tv-vbe [constit cat=ven] |

[constit cat=w] "be" [constit cat=ven];

when vg-pass add [constit cat=vgroup-pass];

// pattern for infinitival verb groups

to-vg := vg-inf;

when to-vg add [constit cat=vgroup-inf];

vg-perf := tv-vbp vg-ven |

[constit cat=w] tv-vbp vg-ven;

when vg-perf add [constit cat=vgroup-perf];

2. Result of the training doc with the otherwise original patterns:

# of matching tags = 88; # of tags in key = 114; # of tags in response = 133;

precision = 66.16%; recall = 77.19%; type accuracy = 100.0%

Result of the test doc with the otherwise original patterns:

# of matching tags = 94; # of tags in key = 130; # of tags in response = 141;

precision = 66.66%; recall = 72.3%; type accuracy = 100.0%

Result of the training doc with the final patterns:

# of matching tags = 94; # of tags in key = 114; # of tags in response = 112;

precision = 83.92%; recall = 82.45%; type accuracy = 100.0%

Result of the test data doc with the final patterns:

# of matching tags = 89; # of tags in key = 130; # of tags in response = 117;

precision = 76.06%; recall = 68.46%; type accuracy = 100.0%

The modfied properties file is below (the one for ariticle2):

# JET properties file

# apply chunkPatterns to article.txt

Jet.dataPath = data

Tags.fileName = pos\_hmm.txt

NameTags.fileName = MUCnameHMM.txt

Pattern.fileName1 = chunkPatternsOrigin.txt

JetTest.fileName1 = article2.txt

processSentence = tokenize, tagNames, tagJet, pat(chunks)

WriteSGML.type = ngroup

Both patterns files have only a single line changed which is listed below:

proper-noun := [ENAMEX TYPE=ORGANIZATION]+ | [ENAMEX TYPE=PERSON]+ |

[ENAMEX TYPE=LOCATION]+;