รหัสโครงการ 12P34C003

โปรแกรมตัดคำไทยด้วยเทคนิคการจำแนกประเภท

BEST 2010: การแข่งขันสุดยอดซอฟต์แวร์แบ่งคำภาษาไทย

รายงานฉบับสมบูรณ์

เสนอต่อ

ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ
สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ
กระทรวงวิทยาศาสตร์และเทคโนโลยี
และ
สำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ (องค์การมหาชน)

ได้รับทุนอุดหนุนโครงการวิจัย พัฒนาและวิศวกรรม
โครงการการแข่งขันพัฒนาโปรแกรมคอมพิวเตอร์แห่งประเทศไทย ครั้งที่ 12
ประจำปีงบประมาณ 2552

โดย

นาย วิทวัส จิตกฤตธรรม
นางสาว ธิติมา นุชพิทักษ์
สถาบันเทคโนโลยีนานาชาติสิรินธร

กิตติกรรมประกาศ

ขอขอบคุณ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติและสำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ ที่ให้ทุนสนับสนุนโครงการนี้ เพื่อร่วมการแข่งขันพัฒนาโปรแกรมคอมพิวเตอร์แห่งประเทศไทยครั้งที่ 12

ขอขอบคุณ คุณณัฐพงศ์ ทองเทพ ที่อนุญาตให้นำรายการคำบ่งบอกตำแหน่งบุคคลมาใช้ในโครงการนี้

โครงการนี้ได้รับการสนันสนุนทรัพยากรการคำนวณจาก ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนา วิทยาศาสตร์และเทคโนโลยีแห่งชาติ URL: http://www.lsr.nectec.or.th

บทคัดย่อ (ไทย)

การตัดคำถือเป็นขั้นตอนที่หลีกเลี่ยงไม่ได้สำหรับการประมวลผลภาษาธรรมชาติ โดยเฉพาะอย่างยิ่งกับภาษาไทยซึ่งไม่มีเครื่องหมายระบุขอบเขตของคำ โครงการนี้นำเสนอระบบตัดคำไทยที่มองปัญหาการตัดคำเป็นปัญหาการจำแนกประเภท (classification) ระบบจะมองว่าทุกตัวอักษรสามารถแบ่งได้เป็น 2 ประเภท คือ เป็นจุดสิ้นสุดของคำ (E) หรือไม่เป็นจุดสิ้นสุดของคำ (I) ในการทำงานระบบจะอ่านและจำแนกทีละตัวอักษร โดยอาศัยแบบจำลองที่ฝึกฝนมาก่อนแล้ว จุดเด่นของระบบอยู่ที่ตัวจำแนกประเภทซึ่งเกิดจากการรวมตัวจำแนกประเภทหลาย ๆตัวเข้าด้วยกันด้วยวิธี Stacking เป็นการเพิ่มความยืดหยุ่นและลดความผิดพลาดที่อาจเกิดจากการยึดติดของแบบจำลองกับชุดฝึกฝน การทดสอบในขั้นเริ่มต้นกับชุดทดสอบหนึ่งแสนคำพบว่าระบบสามารถทำค่า F-measure ได้ 93.84 %

บทคัดย่อ (อังกฤษ)

Word segmentation is an inevitable process in natural language processing,

especially for non-segmented language like Thai. In this work, we propose a Thai word tokenizer in which word segmentation is transformed into a classification problem. In operation, the system scans and classifies each input character into either a word end (denoted as E) or an intermediate character (denoted as I) using a trained model. The characteristic of this system is the use of combined models based on Stacking technique. This improves the system's overall flexibility and reduces error resulted from model overfitting. Initial evaluation on 100K-word test set showed that the system can achieve an F-measure of 93.84 %.

บทนำ

การตัดคำถือเป็นขั้นตอนแรกและเป็นขั้นตอนที่หลีกเลี่ยงไม่ได้สำหรับการประมวลผลภาษาธรรมชาติ (Natural Language Processing) ไม่ว่าจะเป็นงานด้านการค้นคืนสารสนเทศ การแจงประโยค การสกัดสารสนเทศ การแปลภาษา ล้วนแต่จำเป็นต้องระบุขอบเขตของคำจากข้อความให้ได้ก่อนทั้งสิ้น ทั้งนี้หากการตัดคำในขั้นตอนแรกผิด ขั้นตอนการประมวลผลที่ตามมาก็จะผิดไปด้วย ดังนั้นการตัดคำจึงถือว่าเป็นขั้นตอนที่สำคัญที่สุดในการประมวลผลภาษาก็ว่าได้

จนถึงปัจจุบัน เป็นที่ทราบกันดีว่าการตัดคำไทยสามารถแบ่งได้เป็น 3 วิธีใหญ่ ๆ คือ การตัดคำโดยใช้กฎ [1][2] การตัดคำโดยใช้พจนานุกรม[3][4] และการตัดคำโดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) [5][6] นอกจากนี้ยังมีงานวิจัยอื่น ๆที่ใช้เทคนิคผสมผสานอีกด้วย เช่น ตัดคำทั่วไปที่พบบ่อยโดยใช้พจนานุกรมแต่เปลี่ยนไปใช้การเรียนรู้ของเครื่องเมื่อพบคำที่ไม่รู้จัก

ในไม่กี่ปีที่ผ่านมานี้การใช้เทคนิคการเรียนรู้ของเครื่องในงานตัดคำเป็นที่นิยมมากขึ้นด้วยเหตุที่ว่าเครื่องมือและไลบรารี่ต่าง ๆมีพร้อม อีกทั้งยังให้ผลดีเทียบเท่าได้กับการใช้กฎการแบ่งคำที่เขียนโดยนักภาษาศาสตร์ เทคนิคที่นิยมใช้กันมากคือเทคนิคที่มีพื้นฐานมาจาก

Markov chain เช่น N-gram, HMM, CRF ซึ่งทำงานได้ดีกับข้อมูลที่เป็นลำดับ (sequence)

จากการที่แบบจำลองแบบ Markov process ได้ถูกนำมาประยุกต์ใช้อย่างได้ผลดีในหลายงานที่ผ่านมาทำให้เกิดคำถามว่า แบบจำลองแบบที่ใช้สำหรับการจำแนกประเภทแบบอื่น ๆ จะสามารถนำมาประยุกต์กับงานตัดคำได้ดีแค่ไหนเมื่อเทียบกับแบบจำลองอย่าง CRF หรือ HMM คำถามนี้เป็นหนึ่งในจุดเริ่มที่สำคัญของโครงการนี้

สารบัญ

วัตถุประสงค์และเป้าหมาย

- เพื่อส่งเสริมงานวิจัยไทยในด้านการประมวลผลภาษาธรรมชาติ
- เพื่อสร้างเครื่องมือตัดคำไทยเสรีที่ทุกคนสามารถดาวน์โหลดไปใช้ได้โดยไม่เสียค่าใช้จ่าย
- เพื่อทดลองแนวคิดการประยุกต์ใช้วิธีการจำแนกประเภทกับข้อมูลที่เป็นลำดับ

รายละเอียดของการพัฒนา

หลักการ

ระบบที่นำเสนอนี้ใช้เทคนิคการเรียนรู้ของเครื่องแบบหลาย ๆ ตัวประกอบกันเพื่อช่วยในการตัดคำ โดยมองปัญหาการตัดคำเป็นปัญหาการจำแนกประเภท ระบบจะมองว่าทุกตัวอักษรสามารถแบ่งได้เป็น 2 ประเภท คือ เป็นจุดสิ้นสุดของคำ (กำหนดให้เป็น E) หรือไม่เป็นจุดเริ่มของคำ (กำหนดให้เป็น I)[13] ในการทำงานระบบจะอ่านทีละตัวอักษรจากข้อความที่ต้องการตัด แล้วสร้าง feature vector ของแต่ละตัวอักษรโดยดูบริบทรอบ ๆ ประกอบ (คล้ายกับงานของ [8]) feature vector ที่สร้างได้จะถูกส่งไปให้ตัวจำแนกประเภท (classifier) ที่ถูกฝึกสอนมาแล้วเพื่อแบ่งประเภทว่า feature vector นี้เป็น E หรือ I หากเป็น E หมายความว่าตัวอักษรนั้นเป็นจุดสิ้นสุดของคำแล้ว สามารถใส่เครื่องหมาย | เพื่อกำกับขอบเขตของคำได้ หากเป็น I แปลว่าตัวอักษรเป็นส่วนหนึ่งของคำ แต่ไม่ใช่จุดสิ้นสุด

จุดเด่นของระบบที่เสนออยู่ที่ตัวจำแนกประเภทซึ่งมาจากการรวมกันของหลาย ๆตัวจำแนกประเภทโดยใช้วิธี Stacked Generalization [9] หรือ Stacking ซึ่งเป็นวิธีหนึ่งในการรวมตัวจำแนกประเภทหลาย ๆตัวเข้าด้วยกัน โดยค่าความมั่นใจ จากตัวจำแนกประเภทหลาย ๆ ตัวที่ถูกฝึกฝนมาแล้ว (Level-0 Model) จะถูกนำมาสร้างเป็น feature vector อีกอันหนึ่ง feature vector อันใหม่นี้จะถูกส่งไปให้ตัวจำแนกประเภทระดับบน (Level-1

Generalizer) เพื่อแยกประเภทและตัดสินอีกครั้งหนึ่งว่าตัวอักษรที่ระบบกำลังพิจารณาอยู่นั้นควรเป็น E หรือ I

ทฤษฎีที่ใช้

ทฤษฎีหลักที่ใช้ในโครงการนี้คือ Stacked Generalization [9] เสนอโดย David H. Wolpert ซึ่งเป็นวิธีหนึ่งในการรวมผลลัพธ์จากหลาย ๆ ตัวจำแนกประเภทเข้าด้วยกันผ่านตัวจำแนกประเภทระดับบน การทำงานของ Stacked Generalization สามารถนิยามได้ดังนี้

กำหนดชุดฝึกสอน $D=\{(X_i, y_i); i=1,2,...,n\}$ โดยที่ $X=(x_1, x_2, ..., x_f)$ เป็น feature vector ที่มี f มิติ (มี f คุณสมบัติ) ซึ่งเกิดจากการพิจารณาตัวอักษรหนึ่ง ๆ ในข้อความที่ต้องการตัดคำและ $y_i \in \{E, I\}$ เป็นผลลัพธ์เพื่อบอกว่าควรตัดคำหรือไม่

- E หมายความว่าตัวอักษรนั้นเป็นจุดสิ้นสุดของคำ
- I หมายถึงตัวอักษรเป็นส่วนประกอบของคำ ไม่ใช่จุดสิ้นสุดของคำ

กำหนดให้ $M=\{m_1, m_2, ..., m_l\}$ เป็นเซตของโมเดลทั้งหมดที่ได้จากการฝึกสอนโดยใช้คลังข้อความที่มีการกำกับขอบเขตของคำ กำหนดให้ผลลัพธ์การทำนายเวคเตอร์ X บนโมเดลที่ j เป็น $z=m_j(X)$ ซึ่งค่า z นี้อาจเป็นจำนวนจริงหรืออาจเป็นคำตอบ {E, I} เลยก็ได้ขึ้นอยู่กับว่าโมเดลที่ใช้ทำนายทำ classification หรือทำ regression (z อาจใช้ค่า confidence ของการทำนายจากแบบจำลอง classification ก็ได้)

การทำ Stacked Generalization บนโมเดลทั้งหมดใน M คือการสร้างโมเดลระดับบนอีกหนึ่งตัวเรียกว่า M_{Level1} ที่ฝึกสอนโดยใช้ชุดฝึกสอนซึ่งได้มาจากผลลัพธ์ของทุก ๆ โมเดลใน M ซึ่งชุดฝึกสอนดังกล่าวนี้ก็คือ $D_{Level1}=\{(X'_i, y_i); i=1,2,...,n\}$ โดยที่ $X'_i=(m_1(X_i), m_2(X_i), ..., m_l(X_i))$ และ $y_i \in \{E, I\}$ เราเรียกโมเดลใน M ว่า “Level-0 Model” และเรียกโมเดลระดับบน M_{Level1} ว่า “Level-1 Generalizer”

จะเห็นได้ว่าโมเดลสุดท้ายที่ได้คือ M_{Level1} ซึ่งทำหน้าที่ทำนายผลลัพธ์ว่าควรเป็น E

หรือ I จากผลลัพธ์ของ Level-0 Model ทั้งหมด การทำนาย 2 ชั้นแบบนี้ช่วยเพิ่มความยืดหยุ่นในการทำนายและเป็นการปกปิดข้อบกพร่องของแต่ละโมเดลอีกด้วย เพราะโมเดล Level-1 Generalizer สามารถเลือกได้ว่าควรเชื่อผลลัพธ์จากโมเดลไหนมากกว่ากัน ขึ้นอยู่กับบริบทของตัวอักษรที่กำลังพิจารณาอยู่ หลักการนี้คล้ายกับการทำงานของ Multi-layer Neural Network

การประยุกต์ใช้ Stacked Generalization ในโครงการ

ตัวจำแนกประเภทที่นำมาสร้างเป็น Level-0 Model สามารถเป็นได้หลากหลาย แต่ในโครงการนี้ผู้พัฒนาเลือกใช้ decision tree เป็นหลักเพราะทำงานเร็ว แบบจำลองมีขนาดเล็ก และมีความถูกต้องสูง (อ่านคำอธิบายเพิ่มเติมที่ภาคผนวก)

สำหรับ feature จำนวน f ตัวที่เลือกใช้เพื่อเป็นข้อมูลของ $X=(x_1, x_2, ..., x_f)$ นั้น มีทั้งสิ้น 32 feature (f=32) กำหนดให้บริบทปัจจุบันเป็น $...c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}...$ โดย c เป็นตัวอักษรในข้อความและ c_i คือตัวอักษรที่กำลังพิจารณาอยู่ feature ที่ใช้มีดังต่อไปนี้

1. ประเภทของตัวอักษรที่กำลังพิจารณาอยู่ (แทนด้วย $type(c_i)$) ซึ่งดัดแปลงมาจาก [13] ดังตารางข้างล่าง

ประเภทตัวอักษร	คำอธิบาย	ตัวอย่าง
c	อักษรที่สามารถเป็นตัวสะกดได้	กขคฆงจชซญฐดตถท ธนบปพฟภมยรลวศสอ
n	อักษรที่ไม่สามารถเป็นตัวสะกดได้	ฅ ฉ ผ ฝ ฌ ห ฮ ฤ ฦ
r	พยัญชนะที่ไม่ค่อยเริ่มคำ	ฏ ฎ ฑ ฒ ฬ ษ ณ ฃ
v	สระที่ไม่สามารถเริ่มคำ	ะ ิ ี ึ ื ุ ู ั า ำ ๅ ็

w	สระที่สามารถเริ่มคำได้	เ แ โ ใ ไ
t	วรรณยุกต์	อ่ อ้ อ๊ อ๋
s	สัญลักษณ์	อ์ ๆ ฯ . ,
d	ตัวเลข	0-9, ๑-๙
q	เครื่องหมายคำพูด	' “ ({ [ ] })
p	วรรค	
f	ตัวอักษรภาษาต่างประเทศ (เฉพาะตัวอักษร ไม่รวมสัญลักษณ์)	A-Z a-z
o	ตัวอักษรอื่น ๆทั้งหมด	
e	ใช้แทนในกรณีที่ไม่มีตัวอักษรในตำแหน่งนั้น ๆ เช่น ก่อนจุดเริ่มหรือหลังจุดจบของเอกสาร	

2. $type(c_{i-6})$
3. $type(c_{i-5})$
4. $type(c_{i-4})$
5. $type(c_{i-3})$
6. $type(c_{i-2})$
7. $type(c_{i-1})$
8. $type(c_{i+1})$
9. $type(c_{i+2})$
10. $type(c_{i+3})$

11. $type(c_{i+4})$
12. $type(c_{i+5})$
13. $type(c_{i+6})$
14. จำนวนตัวอักษรจาก c_i ไปถึงเครื่องหมายวรรคตัวถัดไป
15. จำนวนตัวอักษรจากเครื่องหมายวรรคล่าสุดที่ผ่านมาถึง c_i
16. สัดส่วนของคำในพจนานุกรมที่ลงท้ายด้วย $c_{i-3}, c_{i-2}, c_{i-1}, c_i$ พจนานุกรมนี้สร้างจากคำทั้งหมดในคลังข้อความของ BEST
17. สัดส่วนของคำในพจนานุกรมที่ลงท้ายด้วย c_{i-2}, c_{i-1}, c_i
18. สัดส่วนของคำในพจนานุกรมที่ลงท้ายด้วย c_{i-1}, c_i
19. สัดส่วนของคำในพจนานุกรมที่ขึ้นต้นด้วย c_{i+1}, c_{i+2}
20. สัดส่วนของคำในพจนานุกรมที่ขึ้นต้นด้วย $c_{i+1}, c_{i+2}, c_{i+3}$
21. สัดส่วนของคำในพจนานุกรมที่ขึ้นต้นด้วย $c_{i+1}, c_{i+2}, c_{i+3}, c_{i+4}$
22. จำนวนตัวอักษรจาก c_i ไปถึงคำบ่งบอกตำแหน่งของบุคคลตัวถัดไป เช่น แพทย์ หญิง ร.ต.ท เป็นต้น
23. จำนวนตัวอักษรจากคำบ่งบอกตำแหน่งของบุคคลล่าสุดที่ผ่านมาถึง c_i
24. P(สิ้นสุดคำ | $c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i$)
    = N($c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i$, '|') / N($c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i$) เมื่อ N(C) คือจำนวนความถี่ของ C ที่เกิดขึ้นในคลังข้อความของ BEST
25. P(สิ้นสุดคำ | $c_{i-3}, c_{i-2}, c_{i-1}, c_i$)
26. P(สิ้นสุดคำ | c_{i-2}, c_{i-1}, c_i)
27. P(สิ้นสุดคำ | c_{i-1}, c_i)
28. P(สิ้นสุดคำ | $c_{i+1}, c_{i+2}, c_{i+3}, c_{i+4}, c_{i+5}$) คล้ายกับ feature ข้างต้นแต่ใช้ความน่าจะ

เป็นแบบมีเงื่อนไขที่ขึ้นกับตัวอักษรด้านขวา

29. P(สิ้นสุดคำ | $c_{i+1}, c_{i+2}, c_{i+3}, c_{i+4}$)

30. P(สิ้นสุดคำ | $c_{i+1}, c_{i+2}, c_{i+3}$)

31. P(สิ้นสุดคำ | c_{i+1}, c_{i+2})

32. N ($c_{i-2}, c_{i-1}, c_i,$ | $, c_{i+1}, c_{i+2}$) / N ($c_{i-2}, c_{i-1}, c_i, *, c_{i+1}, c_{i+2}$) เมื่อ * คือตัวอักษรใด ๆที่เกิดขึ้นใน corpus คุณสมบัตินี้คือความน่าจะเป็นที่จะมี | อยู่หลังตัวอักษรที่พิจารณาอยู่โดยมีเงื่อนไขที่ขึ้นอยู่กับตัวอักษรที่เป็นบริบททั้งฝั่งซ้ายและฝั่งขวา

ข้อมูลความถี่ของตัวอักษรจะถูกเก็บใน Trie ซึ่งนับจากคลังข้อความของ BEST โดยนับความยาวสูงสุดที่ 6 ตัวอักษร (ตัวอักษร 6-grams)

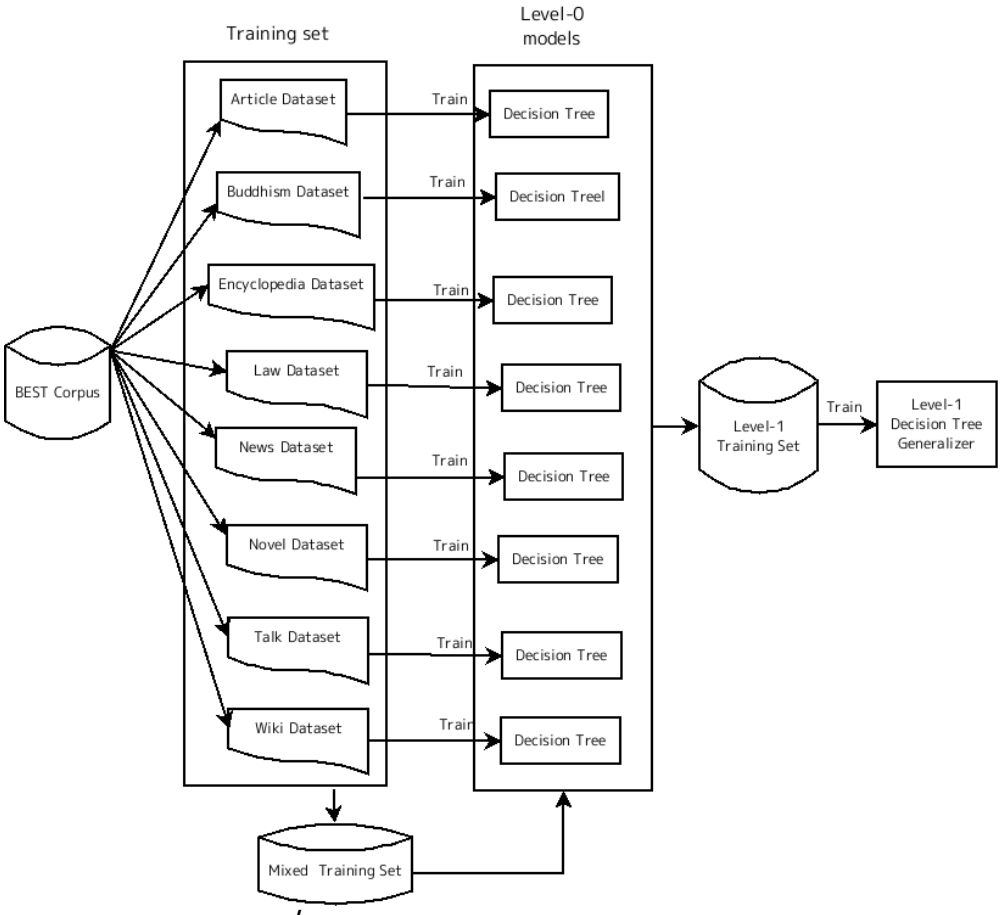
การสร้างแบบจำลอง

การฝึกฝนระบบจะเริ่มจากการสร้างชุดฝึกสอน โดยสร้างจากทุกตัวอักษรที่มีอยู่ใน BEST corpus แต่ละตัวอักษรจะเป็นหนึ่ง feature vector ซึ่งมี feature ตามที่ได้กล่าวไป ชุดฝึกสอนจะถูกแยกเป็น 8 ชุดตาม 8 หมวดหมู่ของบทความที่แจกจ่ายโดยทีมงาน BEST คือ article, buddhism, encyclopedia, law, news, novel, talk และ wiki ตาราง 1 แสดงจำนวนตัวอย่างของแต่ละหมวด

เหตุที่แยกออกเป็นชุด ๆ ตามหมวดเพราะมีเป้าหมายเพื่อนำชุดฝึกสอนจากแต่ละหมวดมาสร้างแบบจำลองหนึ่งตัว ซึ่งทั้งหมดจะได้แบบจำลอง 8 ตัว แบบจำลอง 8 ตัวนี้จะทำหน้าที่เป็นตัวจำแนกประเภทระดับล่างหรือ Level-0 Model ดังรูปที่ 1 จากนั้นชุดฝึกสอนทั้งหมดจะถูกนำมารวมเป็นชุดเดียวแล้วนำมาสอนแบบจำลองระดับบนหรือ Level-1 Generalizer กล่าวโดยรวมคือในระบบนี้จะใช้แบบจำลองทั้งสิ้น 9 ตัว

ตาราง 1: จำนวนตัวอย่างในแต่ละหมวดหมู่ของชุดฝึกสอน

หมวดหมู่	จำนวนตัวอย่าง
article	4487598
buddhism	1832109
encyclopedia	4359980
law	2786863
news	6485132
novel	5568048
talk	1431478
wiki	2994605
รวมทั้งหมด	**29945813**

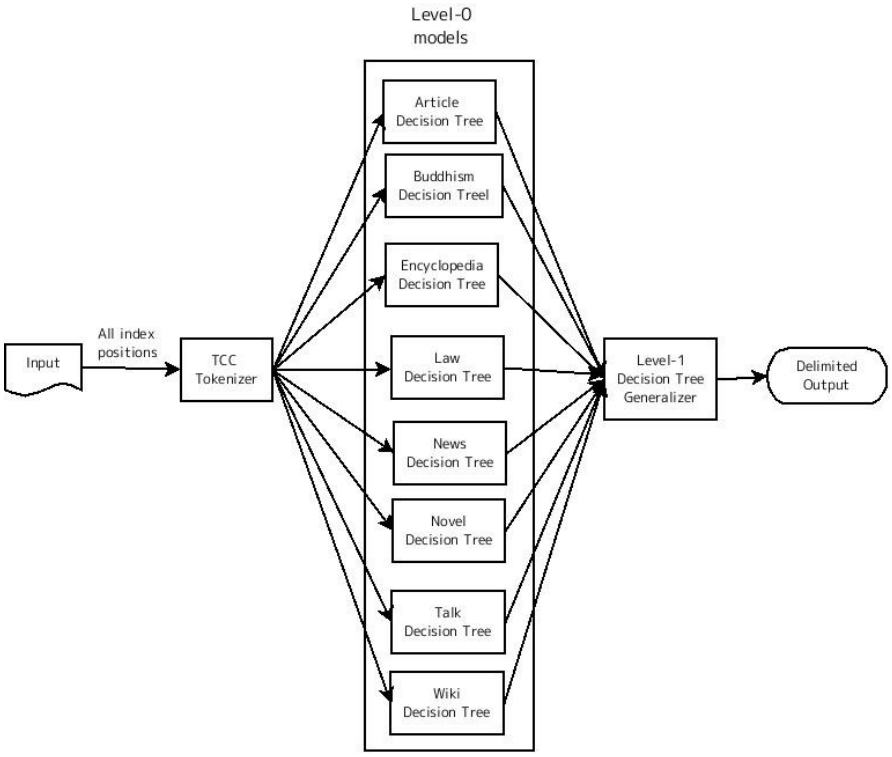


รูปที่ 1: ภาพรวมของการฝึกฝนระบบ

ในการฝึกฝนแบบจำลอง Level-1 หากเป็นไปได้ก็ควรใช้ตัวอย่างทั้งหมดที่มีจากทุกหมวดซึ่งน่าจะให้ผลดีที่สุด แต่เนื่องจากรวมทั้งหมดแล้วมีถึง 29 ล้านตัวอย่าง ซึ่งมากเกินกว่าที่ทรัพยากรการคำนวณที่มีอยู่จะทำได้ ฉะนั้นก่อนการฝึกฝนแบบจำลอง Level-1 จึงต้องทำ Stratified Sampling (การลดปริมาณตัวอย่างลงโดยการหยิบสุ่มในลักษณะที่การกระจายตัวของตัวอย่าง E และตัวอย่าง I ยังคงเดิม) จากแต่ละหมวดมาประมาณ 1.4 ล้านตัวอย่าง รวมกัน 8 หมวดหมู่สุดท้ายการฝึกฝนแบบจำลอง Level-1 จึงใช้ตัวอย่างได้ประมาณ 11.2 ล้านตัวอย่าง

ตามทฤษฎีแบบจำลองที่ใช้เป็น Level-0 Model และ Level-1 Generalizer สามารถเป็นอะไรก็ได้ แต่ในโครงการนี้เลือกที่จะใช้ decision tree เป็นตัวจำแนกประเภทกับทุกหมวด รวมไปถึงตัวจำแนกประเภทระดับบนก็เป็น decision tree ด้วย เหตุที่เลือกใช้เพราะ decision tree สามารถทำงานได้เร็ว แบบจำลองมีขนาดเล็กซึ่งเหมาะกับการนำไปใช้จริง และมีความแม่นยำสูงที่สุดจากผลการทดลอง (ดูภาคผนวก สำหรับข้อมูลการทดลอง)

ขั้นตอนการตัดคำ



การตัดคำจริงจะเริ่มจากการตัด TCC (Thai Character Cluster)[14] จากข้อมูลที่เข้ามาก่อน TCC เป็นกลุ่มของตัวอักษรที่เล็กที่สุดและไม่สามารถแยกจากกันได้ เช่น หากข้อมูลที่จะตัดคำคือ

“เครื่องมือสื่อสารมีหลายชนิด”

เมื่อผ่านตัวตัด TCC (พัฒนาขึ้นใหม่เองด้วยภาษา Java โดยดัดแปลงจาก grammar ที่อธิบายในงานตีพิมพ์) ซึ่งใช้กฎจะได้ผลดังนี้

“เค|รื่อ|ง|มือ|สื่อ|สา|ร|มี|ห|ลา|ย|ช|นิ|ด|”

การใช้ TCC ก่อนใช้แบบจำลองตัดคำเลยมีผลดีคือ สามารถตัดตำแหน่งที่ไม่จำเป็นต้องพิจารณาออกไปได้ทันที เช่น หลัง “เ” เรามั่นใจว่าไม่มี | แน่นอน ตำแหน่งลักษณะแบบนี้ไม่จำเป็นต้องให้แบบจำลองพิจารณาเพราะใช้เวลาช้ากว่าการใช้กฎของ TCC อีกทั้งแบบจำลองยังมีโอกาสทำนายผิดอีกด้วย

เมื่อได้ตำแหน่งของ | จากตัวตัด TCC แล้ว ขั้นต่อไปคือการใช้แบบจำลองเพื่อตัดคำ โดยระบบจะทำนายเฉพาะจุดที่มี | จากตัวตัด TCC เท่านั้น ผลลัพธ์ที่ได้จากการตัดคำโดยแบบจำลองจะเป็นคำตอบคืนแก่ผู้ใช้ การทำงานของระบบอาจมองว่าเป็นการตัดจุดที่เป็นไปไม่ได้ออกทีละขั้นก็ได้ กล่าวคือแทนที่ระบบจะมองว่าข้อมูลที่ป้อนมาเป็น “เครื่องมือสื่อสารมีหลายชนิด” ระบบมองว่าทุกตำแหน่งมี | ขั้นไว้ทั้งหมด คือ

“เ|ค|ร|ื|่|อ|ง|ม|ื|อ|ส|ื|่|ส|า|ร|ม|ี|ห|ล|า|ย|ช|น|ิ|ด|”

จากนั้นค่อยให้ตัวตัด TCC ลดความเป็นไปได้ลงเหลือเป็น

“เค|รื่อ|ง|มือ|สื่อ|สา|ร|มี|ห|ลา|ย|ช|นิ|ด|”

สุดท้ายแบบจำลองที่ฝึกฝนไว้ก็ถูกนำมาใช้ลดตำแหน่งที่ไม่น่าเป็นไปได้ออกจนเหลือ

“เครื่อง|มือ|สื่อ|สาร|มี|หลาย|ชนิด|”

แล้วจึงคืนผลลัพธ์แก่ผู้ใช้

ขั้นตอนการตัดคำเสริม

ระบบที่พัฒนาขึ้นนี้มีขั้นตอนการตัดคำอีกหนึ่งขั้นตอนนั่นคือใช้พจนานุกรม โดยขั้นตอนนี้จะถูกใช้เป็นลำดับแรก (ก่อน TCC) วิธีการคือหาคำที่เกิดขึ้นในพจนานุกรมทั้งหมดโดยใช้วิธีของ Aho-Corasick จากนั้นส่วนที่เป็นคำนั้น ๆจะไม่ถูกส่งไปให้ TCC และแบบจำลองตัดอีกแล้ว นั้นคือเราเชื่อการตัดคำจากพจนานุกรมเลย ขั้นตอนนี้เหมาะที่จะนำไปใช้กับรายการของนิพจน์ระบุนามที่ยาว ๆ เช่น การไฟฟ้าส่วนภูมิภาค ซึ่งยากที่จะตัดให้ถูกโดยแบบจำลอง ตัวอย่างการทำงานของขั้นตอนนี้คือ กำหนดให้ประโยคที่ต้องการตัดเป็น

“เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาค ของเขตนั้น ๆ เพื่อขออนุญาตใช้ไฟฟ้า ”

เริ่มแรกระบบจะหาคำที่อยู่ในพจนานุกรมก่อนซึ่งได้ผลดังนี้ (คำที่ขีดเส้น)

“**เจ้าของบ้าน**จะต้องไปติดต่อที่**การไฟฟ้านครหลวง** หรือ**การไฟฟ้าส่วนภูมิภาค** ของเขตนั้น ๆ เพื่อขออนุญาตใช้ไฟฟ้า ”

จากนั้นระบบจะไม่ส่งส่วนที่ขีดเส้นใต้ไปประมวลผลต่อเพราะถือว่าเป็นคำที่เกิดในพจนานุกรม ฉะนั้นขั้นตอนต่อไปจะมีแค่ส่วนที่ไม่ได้ขีดเส้นใต้ส่งไปให้ TCC ประมวลผลซึ่งได้ดังนี้

“**เจ้าของบ้าน**|จะ|ต้|อ|ง|ไป|ติ|ด|ต่|อ|ที่|**การไฟฟ้านครหลวง**| |หรือ|**การไฟฟ้าส่วนภูมิภาค**|ข|อ|ง|เข|ต|นั้น| ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

จากนั้นแบบจำลองที่ฝึกฝนมาแล้วจะถูกนำมาใช้เพื่อดูจุดที่ TCC ตัดอีกทีหนึ่งดังที่ได้กล่าวไปแล้ว

ขั้นตอนนี้ถูกมองว่าเป็นขั้นตอนที่ไม่สำคัญนักเนื่องจากหากไม่มีขั้นตอนนี้ค่า F-measure ก็ลดลงแค่ประมาณ 0.4% อีกทั้งจากผลการทดลองพบว่าหากใส่คำที่ไม่เหมาะสมลงไปในพจนานุกรม ไม่ว่าจะเป็นคำยาวหรือคำสั้น อาจทำให้ผลการตัดคำแย่ลงได้ จากการทดสอบคำที่ใส่ในพจนานุกรมควรเป็นนิพจน์ระบุนามที่ยาวระดับหนึ่งเท่านั้นจึงจะให้ผลดี

การใส่คำทั่ว ๆไปที่เกิดบ่อยลงไปด้วยจะทำให้ผลตัดคำแย่ลง[1] ในการใช้งานจริงผู้ใช้สามารถเลือกใช้หรือไม่ใช้ขั้นตอนนี้ก็ได้ และสามารถเพิ่มรายการคำลงไปเองได้ด้วย

เครื่องมือที่ใช้ในการพัฒนา

1. ใช้ภาษา Java ในการพัฒนาบน JDK 1.6.0.14
2. ใช้ Netbeans 6.7.1 เป็นเครื่องมือเขียนโปรแกรม
3. ใช้ Ubuntu 9.04 เป็นระบบปฏิบัติการและระบบทดสอบโปรแกรมที่พัฒนา
4. ใช้ไลบรารี่ Weka [10] สำหรับการพัฒนาตัวจำแนกประเภท Decision Tree และทำการทดลองเลือกแบบจำลอง
5. ใช้ ANTLR ซึ่งเป็น Parser Generator จาก grammar ที่ป้อนให้เพื่อสร้าง TCC Parser

เครื่องมือที่กล่าวมาข้างต้นไม่มีค่าใช้จ่ายใด ๆ เป็น open-source และสามารถหาได้จากอินเตอร์เน็ต เนื่องจากใช้ภาษา Java ในการพัฒนา ระบบตัดคำที่พัฒนาได้สามารถรันได้บนทุก platform

1 ค่า F-measure ของระบบในรายงานนี้เกิดจากการใช้ขั้นตอนนี้ด้วย โดยในพจนานุกรมมีเฉพาะนิพจน์ระบุนามที่เกิดในคลังข้อความของ BEST และมีความยาวอย่างน้อย 5 ตัวอักษร

ผลของการทดสอบโปรแกรม

จากการทดสอบในขั้นต้นกับชุดทดสอบหนึ่งแสนคำโดยใช้เครื่องมือทดสอบที่ทีมงาน BEST จัดเตรียมไว้ให้พบว่าได้ผลดังนี้

	BEST 2009	BEST 2010
F-Measure	95.12529 %	93.84095 %
Recall	119299 / 125848 = 94.79610 %	118462 /125848 = 94.13102 %
Precision	119299 / 124977= 95.45676 %	118462 /126626 = 93.55267 %

ผลต่าง F-measure ของตัววัด 2 ปีมีค่าประมาณ 1.29% นั่นหมายความว่าระบบตัดคำตัดนิพจน์ระบุนามผิด 1.29% ซึ่งถือว่าสามารถจัดการกับนิพจน์ระบุนามได้ค่อนข้างดี ที่เหลืออีกประมาณ 6% จึงเป็นคำผิดที่มาจากภาษาอื่นและคำทั่ว ๆไป

โปรแกรมที่พัฒนาขึ้นยังมีข้อผิดพลาดอยู่บ้าง จากการวิเคราะห์ผลลัพธ์พบว่าโดยรวมแล้ว feature ที่เลือกใช้งานทำให้แบบจำลองสามารถระบุได้ว่าจุดที่พิจารณาอยู่นั้นเป็นนิพจน์ระบุนามหรือไม่ แต่ยังไม่สามารถตัดได้ถูกต้องทั้งหมด เห็นได้จากการตัดชื่อคนส่วนใหญ่จะเป็นลักษณะนี้

- นายสมชาย| เรียนดี|
- นางสมหญิง |เรียนเก่ง|

จะเห็นว่าระบบสามารถวิเคราะห์ได้ว่าเป็นนิพจน์ระบุนามและพยายามไม่ใส่ | หลังคำว่าสมหญิง แต่ก็ยังใส่ | ก่อนคำว่าเรียนเก่งซึ่งไม่ถูกต้อง

โดยรวมแล้วคำที่ผิดมักเป็นคำที่มีอยู่ 3 พยางค์ที่ไม่ว่าจะตัดระหว่างกลางหรือไม่ตัดก็สามารถยอมรับได้ในเชิงความหมาย เช่น เพราะ|ฉะนั้น|, เพราะฉะนั้น| เป็นต้น

ระบบสามารถรองรับข้อมูลที่ไม่เคยเห็นได้ดีพอสมควร จากการลองตัดข้อความในหมวด royalnews (จากเว็บ InterBest) ซึ่งมีคำราชาศัพท์และชื่อบุคคลสำคัญในราชวงศ์เยอะ โดยไม่ใช้พจนานุกรม พบว่าสามารถตัดคำยาว ๆที่ไม่ค่อยเกิดในคลังข้อความได้[2] เช่น

- สมเด็จพระเทพรัตนราชสุดาฯ สยามบรมราชกุมารี|
- เสด็จพระราชดำเนิน|
- สาธารณรัฐประชาธิปไตยเนปาล|

ปัญหาและอุปสรรค

ผู้พัฒนาพบว่า corpus ของ BEST มีข้อผิดพลาดอยู่ โดยเฉพาะอย่างยิ่งเรื่องการกำกับ tag พิเศษ เช่น เปิด tag แต่ไม่ปิด tag </NE> หรือสะกด tag ผิดเป็น /NE> ทำให้การอ่านข้อมูลมาฝึกฝนระบบค่อนข้างจะมีปัญหาจนต้องแก้ไขด้วยตัวเองหลายแห่ง

อุปสรรคอีกอย่างหนึ่งคือปริมาณตัวอย่างที่มากจนไม่สามารถนำมาคำนวณหมดได้ เนื่องจากงานนี้ใช้หนึ่งตัวอักษรเป็นหนึ่งตัวอย่างจึงทำให้มีจำนวนตัวอย่างเท่ากับจำนวนตัวอักษรในคลังข้อความ การฝึกฝนแบบจำลองหรือการทดลองในแต่ละครั้งต้องใช้เวลานานมาก (หลายวันต่อหนึ่งการทดลอง) ผู้พัฒนาต้องขอขอบคุณ หน่วยปฏิบัติการวิจัยการจำลองขนาดใหญ่ ที่อนุญาตให้ใช้งาน cluster และทำให้สามารถทำการทดลองได้สะดวกขึ้น

แนวทางในการพัฒนาต่อ

หัวใจสำคัญของระบบตัดคำที่พัฒนานี้คือ feature ยิ่งมี feature ที่สามารถเป็นปัจจัยช่วยให้ระบบตัดสินได้ว่าจุดที่ดูอยู่ควรตัดหรือไม่ตัดมากเท่าใด ระบบก็ยิ่งดี ฉะนั้นการพัฒนาต่อที่ควรทำคือหา feature เพิ่มเติม เช่น นำ POS จากคลังข้อความ Orchid มาใช้, นอกจากตัวอักษรแล้วก็หา feature เชิงความหมายด้วยก็น่าจะทำให้ตัดได้ดีขึ้น

2 เนื่องจากระบบทดสอบของ BEST ไม่มีให้ทดสอบหมวด royalnews จึงไม่สามารถบอกค่า F-measure ได้

คู่มือการติดตั้ง

1. ติดตั้ง JRE หรือ JDK version ใหม่ล่าสุดจากเว็บ http://java.sun.com/javase/downloads/index.jsp ก่อน
2. เปิดดูไฟล์บนแผ่น CD จะพบ 3 โฟลเดอร์ ให้ copy โฟลเดอร์ชื่อ program ไปไว้ที่ใดก็ได้บนเครื่อง (หรือจะรันจากแผ่นก็ได้)

คู่มือการใช้งาน

1. เข้าไปที่โฟลเดอร์ program ที่ copy ในขั้นตอนที่แล้วโดยใช้ command prompt (Windows) หรือ terminal (Linux)
2. รัน run.bat (สำหรับ Windows) หรือรัน run.sh (สำหรับ Linux) หากโปรแกรมโหลดสำเร็จจะเห็น >> ด้านซ้ายมือ

ตัวอย่างการใช้งาน

หากต้องการตัดคำ “การแข่งขันพัฒนาซอฟท์แวร์” ให้พิมพ์คำสั่ง

- >> การแข่งขันพัฒนาซอฟท์แวร์:tok

หากต้องการตัดคำ “การแข่งขันพัฒนาซอฟท์แวร์” แล้วให้ผลออกไปที่ไฟล์ที่อยู่ที่ C:\a.txt ให้พิมพ์คำสั่ง

- >> การแข่งขันพัฒนาซอฟท์แวร์:tok:C:\a.txt

หากต้องการตัดคำจากเนื้อหาในไฟล์ที่อยู่ที่ C:\b.txt ให้พิมพ์

- >> C:\b.txt:tokfile

หากต้องการตัดคำจากเนื้อหาในไฟล์ที่อยู่ที่ C:\b.txt แล้วให้ผลออกไปที่ C:\out.txt ให้พิมพ์

- >> C:\b.txt:tokfile:C:\out.txt

ดูคำสั่งอื่น ๆโดยพิมพ์

- >> h

เอกสารอ้างอิง

1. David Palmer. A Trainable Rule-based Algorithm for Word Segmentation. 1997.
2. ปโยธร อุราธรรมกุล, กานดา รุณนะพงศา. “การปรับปรุงกฎสำหรับตัดคำในเอกสารไทย” (Improved Rule-Based for Thai Documents). *NECSEC*. 2005
3. Raruenrom Samphan. "การแบ่งคำไทยด้วยพจนานุกรม" (Dictionary-based Thai Word Separation). Senior Project Report. *Department of Computer Engineering, Chulalongkorn University*, Bangkok. 1991.
4. เล็กซ์โต : LexTo. http://sansarn.com/lexto/
5. ทีเล็กส์ : Tlexs. http://www.sansarn.com/tlexs/
6. Choochart Haruechaiyasak, Sarawoot Kongyoung, Chaianun Damrongrat. LearnLexTo: a machine-learning based word segmentation for indexing Thai texts. *Proceeding of the 2nd ACM workshop on Improving non english web searching.* pp. 85-88, 2008

7. Canasai Kruengkrai, Virach Sornlertlamvanich, and Hitoshi Isahara. A Conditional Random Field Framework for Thai Morphological Analysis. P*roceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Poster Sessions, May 24-26, 2006. Genoa, Italy.

8. Thanaruk Theeramunkong, Sasiporn Usanavasin. Non-Dictionary-Based Thai Word Segmentation Using Decision Trees. *Proceedings of the first international conference on Human language technology research*. pp. 1-5, 2001.

9. David H. Wolpert. Stacked Generalization. *Neural Networks*. pp. 241-259, 1992.

10. Ian H. Witten and Eibe Frank. Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

11. Chih-chung Chang, Chih-Jen Lin. LIBSVM: a Library for Support Vector Machines. 2001.

12. CRF++: Yet Another CRF toolkit : http://crfpp.sourceforge.net/

13. Choochart Haruechaiyasak, Sarawoot Kongyoung, Matthew N. Dailey, A Comparative Study on Thai Word Segmentation Approaches. I*n Proceedings of ECTI-CON*, 2008.

14. Thanaruk Theeramunkong, Virach Sornlertlamvanich, Thanasan Tanhermhong, Wirat Chinnan. Character Cluster Based Thai Information Retrieval. 2002.

ภาคผนวก

เหตุผลการเลือกใช้ Decision Tree

ในการเลือกตัวจำแนกประเภทที่จะใช้สร้างเป็นแบบจำลองนั้น ผู้พัฒนาเลือกโดยลองใช้ตัวจำแนกประเภททั้งหมด 8 ตัวกับชุดฝึกสอนทั้ง 8 แบบ (จาก 8 หมวด) ซึ่งจะมีคู่ของตัวจำแนกประเภทกับชุดทดสอบทั้งหมด 64 คู่ เพื่อให้ได้ค่าที่น่าเชื่อถือมากที่สุด ในการทดลองแต่ละคู่จึงใช้วิธี 10-fold cross validation ด้วย ส่งผลให้ 1 คู่มีค่าความถูกต้อง 10 ค่า การสรุปผลจะใช้วิธีหาค่าเฉลี่ยจาก 10 ค่านี้ เพื่อไม่ให้การทดลองกินเวลานานชุดทดสอบจากแต่ละหมวดจะถูกทำ Stratified Sampling ให้เหลือประมาณ 3 หมื่นตัวอย่างก่อนเริ่มการทดลอง

ตัวจำแนกประเภททั้ง 8 ตัวที่ใช้ทดสอบคือ One R, Naïve Bayes, kNN (ตั้ง k=7), Decision Tree, Bayesian Network, RBF Network, Zero R, และ Random Tree ค่า accuracy ที่ได้ (เฉลี่ยจาก 10 รอบของ cross validation) เป็นดังนี้

ชุดทดสอบ	Decision Tree	One R	Naïve Bayes	7-NN	Bayesian Network	RBF Network	Zero R	Random Tree
article	**94.959**	84.072	83.919	87.469	92.360	84.125	73.117	87.375
buddhism	**96.387**	82.812	86.338	89.718	93.857	86.997	70.635	89.188
encyclopedia	**94.815**	84.442	84.911	87.924	92.421	85.200	73.307	86.012
law	**96.534**	84.844	86.309	90.469	93.510	86.501	73.877	91.100
news	**93.612**	83.673	82.194	87.590	91.140	82.447	74.137	85.585
talk	**95.525**	83.927	84.187	88.130	93.341	85.231	71.420	87.312
novel	**95.242**	83.547	84.550	87.022	93.720	84.959	70.180	86.517
wiki	**93.669**	81.466	83.417	87.238	90.627	82.279	73.561	84.412
Average	**95.093**	83.598	84.478	88.195	92.622	84.717	72.529	87.188

จะเห็นว่า decision tree ดีกว่าตัวจำแนกประเภทอื่น ในทุกหมวดหมู่ (อย่างมีนัยสำคัญ) ผู้พัฒนาจึงเลือกใช้ decision tree กับทุกหมวดเพื่อเป็น Level-0 model และ Level-1 Generalizer ด้วย

การทดลองอื่น ๆ

นอกจากการคิดค้น feature ใหม่ ๆแล้วสิ่งที่ผู้พัฒนากำลังทำอยู่คือการทดลอง Algorithm ใหม่ ๆ นอกจาก decision tree ต้นเดียว จากการที่ decision tree ให้ผลค่อนข้างดีในแง่ของ accuracy ของการจำแนกประเภท ผู้พัฒนาจึงนำ algorithm ที่ใช้ tree อื่น ๆ มาลองเพิ่มเติม ตาราง 2 แสดงผลลัพธ์การตัดคำโดยใช้ algorithm ต่าง ๆกับชุดทดสอบหนึ่งแสนคำของ BEST (ผลลัพธ์จากการใช้เครื่องมือประเมินของ BEST) โดยที่โมเดลทุกตัวถูกฝึกกับตัวอย่างจากหมวดละ 20,000 ตัวอย่าง (รวมแปดหมวดเท่ากับ 1.6 แสนตัวอย่าง)

คำอธิบาย

- RF_IxKy หมายถึง Random forest ที่มี decision tree ทั้งหมด x ต้นและแต่ละต้นจะพิจารณา feature แค่ y ตัว
- rss_Px_J48 หมายถึง Random subspace โดยใช้ร่วมกับ J48 (decision tree) โดยพิจารณาจำนวน feature เป็นสัดส่วน x ของ feature ทั้งหมด เช่น x=0.7 หมายความว่าพิจารณา 70% ของจำนวน feature ที่มีอยู่แบบสุ่ม
- NB หมายถึง Naïve Bayes

จะเห็นได้ว่า random forest ให้ค่า F-measure สูงสุดอยู่ที่ 88.022%[3] โดยใช้ทั้งหมด 20 ต้นและดูทีละ 10 feature จากการวิเคราะห์ดูผลลัพธ์จะเห็นว่ายิ่งมีจำนวนต้นไม้มาก (ค่า I) ก็ยิ่งได้ค่า F-measure ที่สูง ไม่ว่าจะใช้กี่ feature ก็ตาม ทำให้เกิดคำถามว่าหากเพิ่ม

3 ค่า F-measure ต่ำกว่าการทำ Stacking ที่รายงานไป แต่การทดลองนี้มิได้ใช้ตัวอย่างเต็มจริง ๆเนื่องจากจะกินเวลานานมาก ทำให้ผลการทดลองนี้ไม่อาจบอกได้ว่า Random Forest ดีกว่า Stacking หรือไม่ บอกได้แต่เพียงว่า Random Forest ให้ผลค่อนข้างดี

	BEST 2009			BEST 2010		
Scheme	Precision	Recall	F-Measure	Precision	Recall	F-Measure
RF_I20K10_20000	90.191	90.527	90.359	86.774	89.307	88.022
RF_I20K20_20000	90.087	90.479	90.282	86.547	89.211	87.859
RF_I20K15_20000	90.085	90.388	90.236	86.556	89.141	87.830
rss_P0.7_J48_20000	90.051	90.395	90.222	86.475	89.110	87.773
RF_I15K10_20000	90.019	90.175	90.097	86.612	88.957	87.769
RF_I15K20_20000	90.016	90.239	90.127	86.449	88.952	87.683
RF_I15K15_20000	90.005	90.121	90.063	86.482	88.864	87.657
RF_I20K5_20000	89.693	90.187	89.940	86.214	88.937	87.554
RF_I10K10_20000	89.571	90.090	89.830	85.984	88.795	87.367
RF_I10K20_20000	89.555	90.152	89.852	85.856	88.797	87.302
RF_I15K5_20000	89.496	89.837	89.666	86.030	88.583	87.288
RF_I10K15_20000	89.449	89.930	89.689	85.809	88.628	87.196
RF_I10K5_20000	89.042	89.868	89.453	85.391	88.521	86.928
RF_I5K10_20000	88.903	88.701	88.802	85.358	87.423	86.378
RF_I5K15_20000	88.896	88.661	88.778	85.311	87.383	86.335
RF_I5K20_20000	88.742	88.400	88.571	85.227	87.130	86.168
j48_20000	88.682	87.743	88.210	85.242	86.545	85.889
RF_I5K5_20000	88.088	88.068	88.078	84.534	86.786	85.645
rss_P0.5_J48_20000	87.537	86.566	87.049	83.924	85.252	84.583
NB_20000	71.367	67.527	69.394	66.226	65.652	65.938

ตาราง 2: ผลลัพธ์การทดลองใช้โมเดลต่าง ๆกับชุดทดสอบหนึ่งแสนคำ

ต้นไม้เข้าไปอีก ให้มากกว่า 20 ต้นจะได้ค่า F-measure สูงขึ้นอีกหรือไม่ ตาราง 3 แสดงผลการทดลองเพิ่มเติมนี้ โดยครั้งนี้เพิ่มตัวอย่างเป็นหมวดหมู่ละ 30,000 ตัวอย่าง (รวม 8 หมู่ ได้ 2.4 แสนตัวอย่าง)

จะผลการทดลองใหม่นี้พบว่าการเพิ่มต้นไม้เข้าไปใน Random Forest มากกว่า 20 ต้นไม่เป็นการเพิ่มประสิทธิภาพ (ไม่ว่าจะใช้กี่ feature) นอกจากแบบจำลองจะใหญ่มาก (เกิน 100 MB) แล้วยังทำให้ F-measure ลดลงอีกด้วย

	BEST 2009			BEST 2010		
Scheme	Precision	Recall	F-Measure	Precision	Recall	F-Measure
RF_I30K10_30000	89.511	87.649	88.571	87.747	87.067	87.406
RF_I30K20_30000	89.414	87.522	88.458	87.557	86.911	87.233
RF_I25K10_30000	89.414	87.296	88.342	87.666	86.725	87.193
RF_I20K10_30000	89.253	87.406	88.320	87.508	86.831	87.168
RF_I25K20_30000	89.386	87.366	88.365	87.553	86.762	87.156
rss_P0.6_J48_30000	89.176	87.211	88.182	87.559	86.697	87.126
RF_I20K20_30000	89.197	87.295	88.236	87.373	86.701	87.035
RF_I30K30_30000	89.167	87.097	88.120	87.350	86.511	86.928
rss_P0.7_J48_30000	89.299	86.767	88.015	87.622	86.215	86.913
RF_I20K30_30000	89.120	87.134	88.116	87.289	86.538	86.912
RF_I25K30_30000	89.136	86.903	88.005	87.315	86.311	86.810
rss_P0.8_J48_30000	88.997	86.079	87.514	87.208	85.484	86.338

ตาราง 3: ผลการทดลองใช้โมเดลแบบต้นไม้โดยทดลองปรับค่าพารามีเตอร์ต่าง ๆ

สำหรับกรณีของ Random Subspace พบว่ายิ่งลดปริมาณของ feature ที่ใช้พิจารณายิ่งให้ผลดีขึ้น จุดนี้ตรงกับ Random Forest ตรงที่ว่ายิ่งเพิ่มปริมาณ feature ที่พิจารณาไม่ได้ทำให้ได้ผลดีขึ้น เราสามารถพอสรุปอย่างคร่าว ๆได้ ณ ที่นี้ว่าปริมาณ feature ที่เหมาะสมกับ algorithm ที่ใช้ต้นไม้ที่มีการดู subspace ย่อยคือประมาณ 10 feature การทดลองยังคงต้องดำเนินต่อไปก่อนที่จะสามารถสรุปได้ว่าโมเดลใดเหมาะสมที่สุด