Unsupervised Morphological Learning for Bangla

Sajib Dasgupta and Vincent Ng

Human Language Technology Research Institute

University of Texas at Dallas

Morphological Analysis / Word Segmentation

- Segment a word into morphemes (roots, prefixes, suffixes)

Morphological Analysis / Word Segmentation

- Segment a word into morphemes (roots, prefixes, suffixes)

- **English:**

  “unforgettable”

  = “un” (Prefix) + “forget” (Root) + “able” (Suffix)

- **Bangla:**

  “অনাধুনিকতার” (anAdhUnIkTAr)

  = “an” (Prefix) + “@dhUnIk” (Root) + “TA” (Suffix) + “r” (Inflectional Suffix)

Why Morphological Analysis?

- Stemming
- POS tagging
- High-level NLP applications (e.g., text classification)

Why Morphological Analysis?

- Stemming
- POS tagging
- High-level NLP applications (e.g., text classification)

Morphological Analysis: Two Major Approaches

- Knowledge-based approaches
  - Rely on manually designed segmentation heuristics

Morphological Analysis: Two Major Approaches

- Knowledge-based approaches
  - Rely on manually designed segmentation heuristics
- Unsupervised approaches
  - Induce morphemes from a large, unannotated corpus

Morphological Analysis: Two Major Approaches

- Knowledge-based approaches
  - Rely on manually designed segmentation heuristics

- Unsupervised approaches
  - Induce morphemes from a large, unannotated corpus
  - Successfully applied to many European languages such as English, German, and Dutch (e.g., Goldsmith (2001), Schone and Jurafsky (2001), Freitag (2005))

Morphological Analysis: Two Major Approaches

- Knowledge-based approaches
  - Rely on manually designed segmentation heuristics

- Unsupervised approaches
  - Induce morphemes from a large, unannotated corpus
  - Successfully applied to many European languages such as English, German, and Dutch (e.g., Goldsmith (2001), Schone and Jurafsky (2001), Freitag (2005))
  - Not so successful for agglutinative languages such as Finnish and Turkish (see 2006 PASCAL Challenge on Unsupervised Segmentation of Words into Morphemes)

Goal

Unsupervised morphological analysis for Bangla

Goal

Unsupervised morphological analysis for Bangla

How difficult is morphological parsing of Bangla?

Goal

Unsupervised morphological analysis for Bangla

How difficult is morphological parsing of Bangla?

- Bangla is highly inflectional but not agglutinative
- More difficult than English
- Less difficult than Turkish and Finnish

Our Unsupervised Word Segmentation Algorithm

1. Morpheme induction
   - Induce morphemes from a vocabulary V (a list of words taken from a large, unannotated corpus)

2. Segmentation
   - Segment a word based on the induced morphemes

Our Unsupervised Word Segmentation Algorithm

1. Morpheme induction
   - Induce morphemes from a vocabulary V (a list of words taken from a large, unannotated corpus)

2. Segmentation
   - Segment a word based on the induced morphemes

Our Unsupervised Word Segmentation Algorithm

1. Morpheme induction
   - Induce morphemes from a vocabulary V (a list of words taken from a large, unannotated corpus)

2. Segmentation
   - Segment a word based on the induced morphemes

The Morpheme Induction Algorithm

- Basic morpheme induction method

- Three extensions to the basic induction method

The Morpheme Induction Algorithm

- Basic morpheme induction method
- Three extensions to the basic induction method

Basic Prefix and Suffix Induction Method

- Motivated by Keshava and Pitler’s (2006) algorithm
- Let A and B be two character sequences. Assume:
  1. AB and A in V $\Rightarrow$ B is a suffix

Basic Prefix and Suffix Induction Method

- Motivated by Keshava and Pitler’s (2006) algorithm
- Let A and B be two character sequences. Assume:
  1. AB and A in V $\Rightarrow$ B is a suffix
     - “singing” and “sing” $\Rightarrow$ “ing” is a suffix

Basic Prefix and Suffix Induction Method

- Motivated by Keshava and Pitler's (2006) algorithm
- Let A and B be two character sequences. Assume:
  1. AB and A in V $\Rightarrow$ B is a suffix
     - "singing" and "sing" $\Rightarrow$ "ing" is a suffix
  2. AB and B in V $\Rightarrow$ A is a prefix

Basic Prefix and Suffix Induction Method

§ Motivated by Keshava and Pitler's (2006) algorithm

§ Let A and B be two character sequences. Assume:

1. AB and A in V $\Rightarrow$ B is a suffix
    - "singing" and "sing" $\Rightarrow$ "ing" is a suffix
2. AB and B in V $\Rightarrow$ A is a prefix
    - "preset" and "set" $\Rightarrow$ "pre" is a prefix

Basic Prefix and Suffix Induction Method (Cont')

Problem: Assumption does not always hold

- "diverge" and "diver" are in V "ge" is a suffix Wrong!

Basic Prefix and Suffix Induction Method (Cont')

Problem: Assumption does not always hold

- "diverge" and "diver" are in V "ge" is a suffix Wrong!

  Many of the induced prefixes and suffixes are erroneous

Basic Prefix and Suffix Induction Method (Cont')

Problem: Assumption does not always hold

- "diverge" and "diver" are in V "ge" is a suffix Wrong!

  Many of the induced prefixes and suffixes are erroneous

Solution: score each induced affix and retain only those whose scores are above a pre-defined threshold

- Score(a) = affix-frequency(a) * length(a)

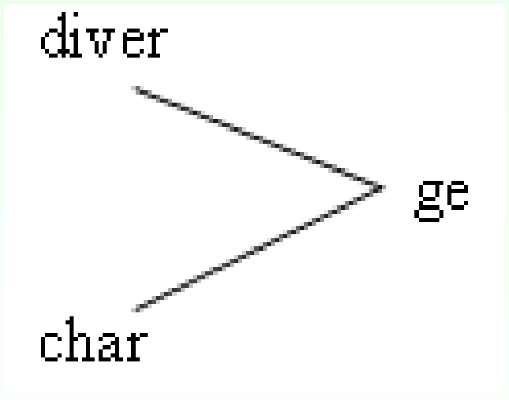
Affix Frequency

- Number of distinct words in V to which an affix attaches

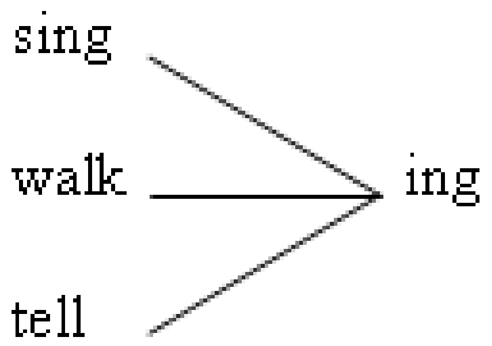
Affix Frequency

- Number of distinct words in V to which an affix attaches

- Affix frequency of "ge" = 2



- Affix frequency of "ing" = 3



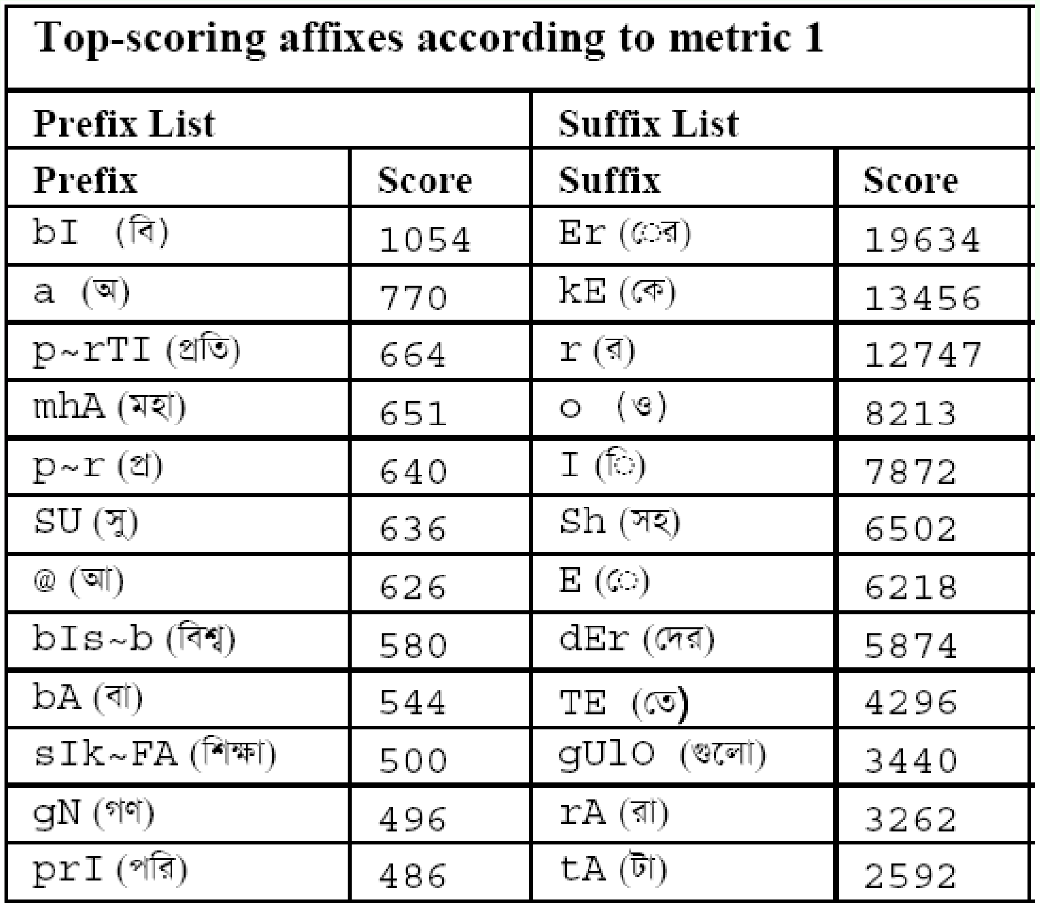
Why Should the Score of an Affix Depend on its Affix Frequency?

- The higher the affix frequency
  - The more words to which the affix attaches
  - The more likely the affix is correct

Why Should the Score of an Affix Depend on its Length?

- Shorter affixes are more likely to be incorrect than longer affixes (Goldsmith (2001))
  - A higher score should be given to a longer affix

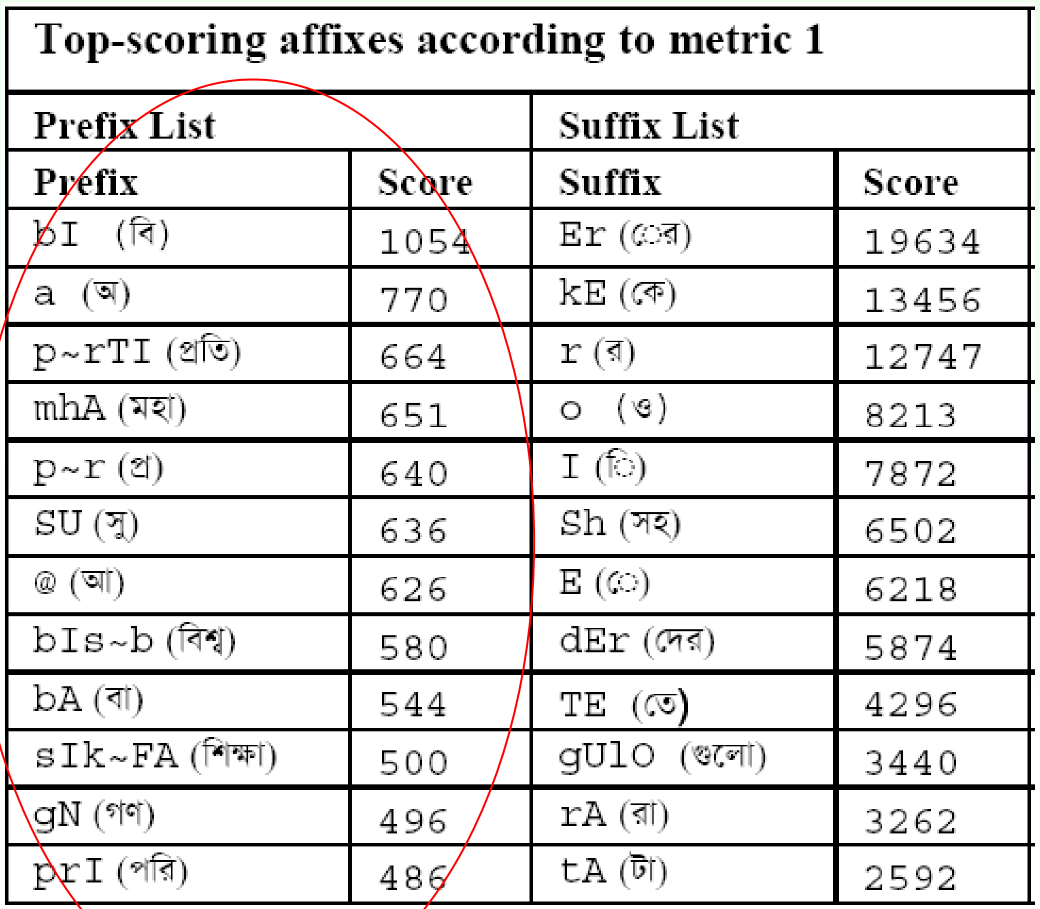
Top Scoring Affixes According to the Metric



Top-scoring affixes according to metric 1

Prefix List		Suffix List	
Prefix	Score	Suffix	Score
bI (বি)	1054	Er (ের)	19634
a (অ)	770	kE (কে)	13456
p~rTI (প্রতি)	664	r (র)	12747
mhA (মহা)	651	o (ও)	8213
p~r (প্র)	640	I (ি)	7872
SU (সু)	636	Sh (সহ)	6502
@ (আ)	626	E (ে)	6218
bIs~b (বিশ্ব)	580	dEr (দের)	5874
bA (বা)	544	TE (তে)	4296
sIk~FA (শিক্ষা)	500	gUlO (গুলো)	3440
gN (গণ)	496	rA (রা)	3262
prI (পরি)	486	tA (টা)	2592

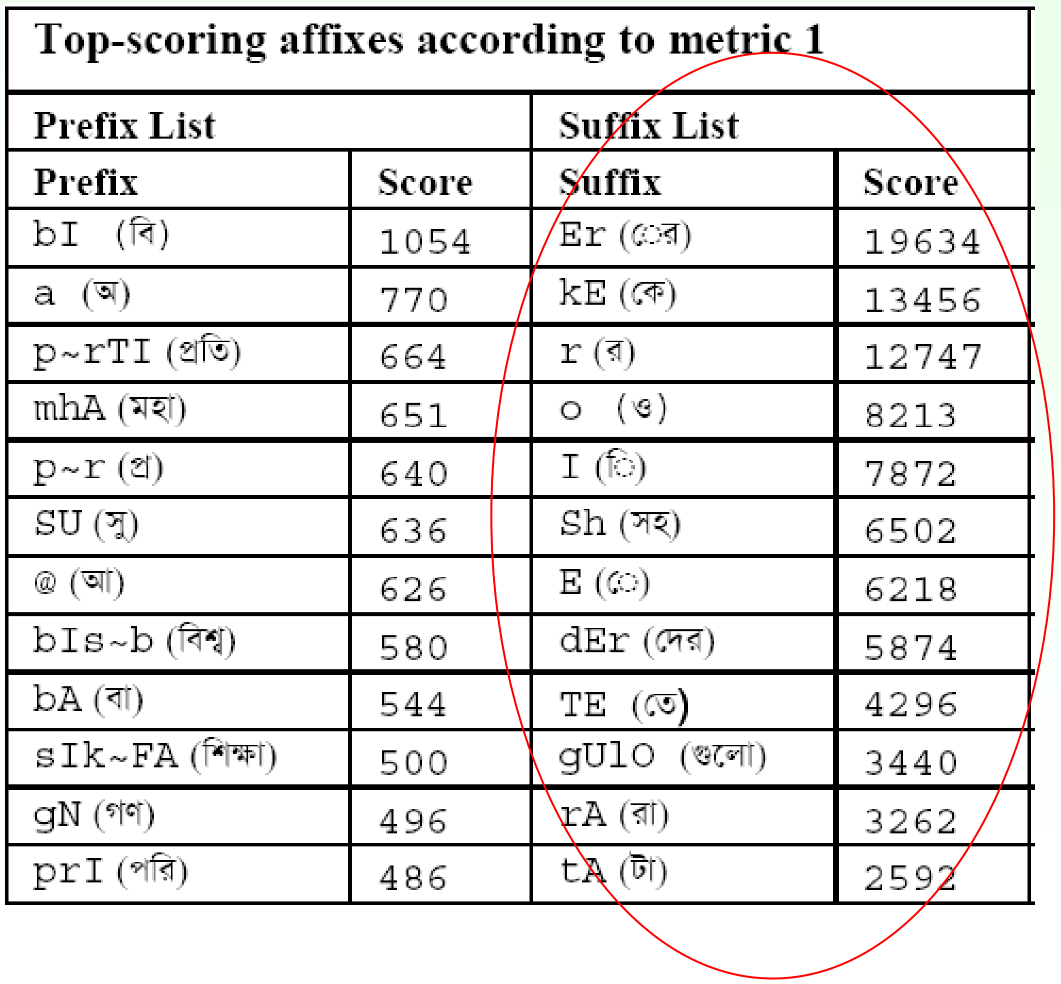
Top Scoring Affixes According to the Metric



Top-scoring affixes according to metric 1

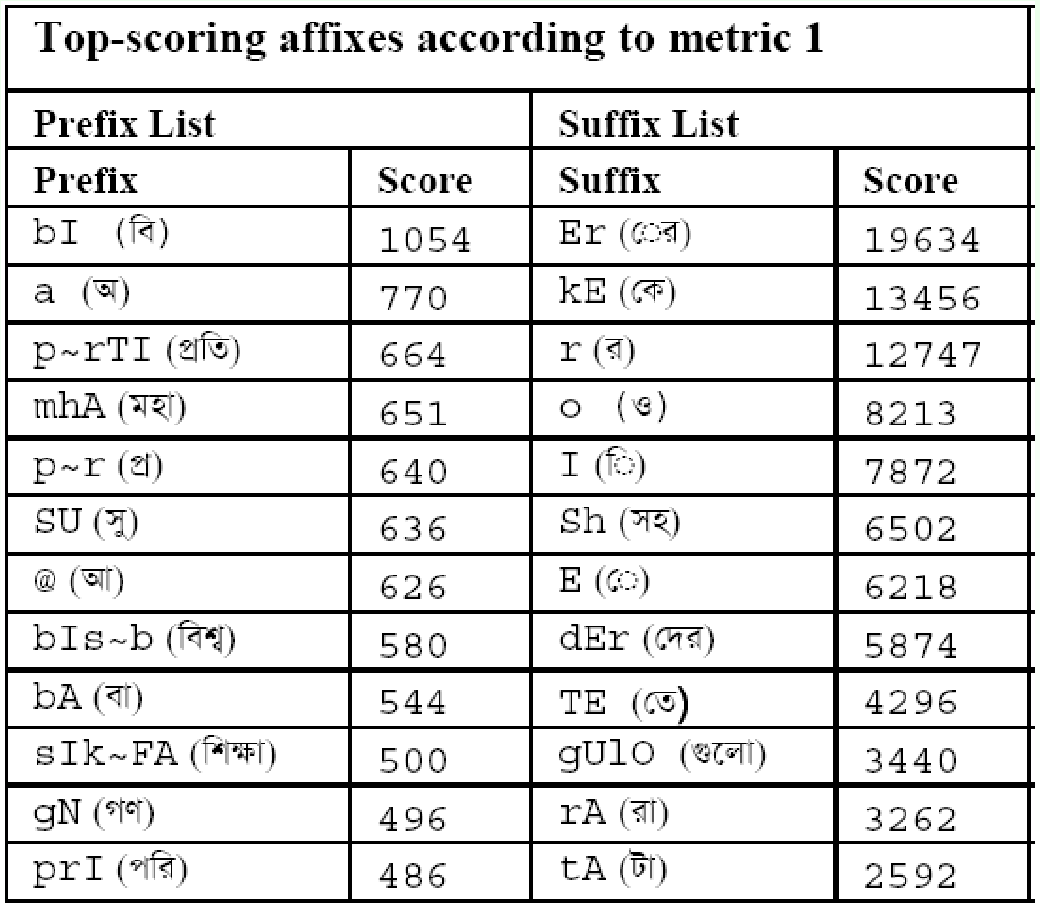
Prefix List		Suffix List	
Prefix	Score	Suffix	Score
bI (বি)	1054	Er (ের)	19634
a (অ)	770	kE (কে)	13456
p~rTI (প্রতি)	664	r (র)	12747
mhA (মহা)	651	o (ও)	8213
p~r (প্র)	640	I (ি)	7872
SU (সু)	636	Sh (সহ)	6502
@ (আ)	626	E (ে)	6218
bIs~b (বিশ্ব)	580	dEr (দের)	5874
bA (বা)	544	TE (তে)	4296
sIk~FA (শিক্ষা)	500	gUlo (গুলো)	3440
gN (গণ)	496	rA (রা)	3262
prI (পরি)	486	tA (টা)	2592

Top Scoring Affixes According to the Metric



Top-scoring affixes according to metric 1			
Prefix List		Suffix List	
Prefix	Score	Suffix	Score
bI (বি)	1054	Er (ের)	19634
a (অ)	770	kE (কে)	13456
p~rTI (প্রতি)	664	r (র)	12747
mhA (মহা)	651	o (ও)	8213
p~r (প্র)	640	I (ি)	7872
SU (সু)	636	Sh (সহ)	6502
@ (আ)	626	E (ে)	6218
bIs~b (বিশ্ব)	580	dEr (দের)	5874
bA (বা)	544	TE (তে)	4296
sIk~FA (শিক্ষা)	500	gUlO (গুলো)	3440
gN (গণ)	496	rA (রা)	3262
prI (পরি)	486	tA (টা)	2592

Top Scoring Affixes According to the Metric



Top-scoring affixes according to metric 1

Prefix List		Suffix List	
Prefix	**Score**	**Suffix**	**Score**
bI (বি)	1054	Er (ের)	19634
a (অ)	770	kE (কে)	13456
p~rTI (প্রতি)	664	r (র)	12747
mhA (মহা)	651	o (ও)	8213
p~r (প্র)	640	I (ি)	7872
SU (সু)	636	Sh (সহ)	6502
@ (আ)	626	E (ে)	6218
bIs~b (বিশ্ব)	580	dEr (দের)	5874
bA (বা)	544	TE (তে)	4296
sIk~FA (শিক্ষা)	500	gUlO (গুলো)	3440
gN (গণ)	496	rA (রা)	3262
prI (পরি)	486	tA (টা)	2592

Scoring an Affix

- We retain an affix in the induced list if and only if its score exceeds the pre-defined threshold
  - 60 for prefixes and 40 for suffixes

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction

- Three improvements to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

Basic Root Induction Method

- Now, we have a list of induced prefixes and suffixes.
- For each word w in V, check whether w is divisible based on this affix list.

Basic Root Induction Method

- Now, we have a list of induced prefixes and suffixes.
- For each word w in V, check whether w is divisible based on this affix list.
  - If w = “r + s” or “p + r”, where p and s are an induced prefix and suffix respectively and r is another word in the vocabulary, then w is divisible and so w can’t be a root.

Basic Root Induction Method

- Now, we have a list of induced prefixes and suffixes.
- For each word w in V, check whether w is divisible based on this affix list.
  - If w = “r + s” or “p + r”, where p and s are an induced prefix and suffix respectively and r is another word in the vocabulary, then w is divisible and so w can’t be a root.
  - If not, then we add w to the list of candidate roots.

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction

- Three extensions to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction
- Three extensions to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction
- Three extensions to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

Employing Length-Dependent Thresholds

- Recall that we retain an induced affix in our list if and only if its score exceeds some threshold
  - 60 for prefixes and 40 for suffixes
  - Threshold is independent of the length of an affix

Employing Length-Dependent Thresholds

- Recall that we retain an induced affix in our list if and only if its score exceeds some threshold
  - 60 for prefixes and 40 for suffixes
  - Threshold is independent of the length of an affix

- since shorter affixes are more likely to be erroneous, we hypothesize that employing larger thresholds for shorter affixes may yield better performance

Employing Length-Dependent Thresholds

- Recall that we retain an induced affix in our list if and only if its score exceeds some threshold
  - 60 for prefixes and 40 for suffixes
  - Threshold is independent of the length of an affix

- since shorter affixes are more likely to be erroneous, we hypothesize that employing larger thresholds for shorter affixes may yield better performance
  - If affix length < 2, multiply threshold by (4 – affix length)

Employing Length-Dependent Thresholds

- Recall that we retain an induced affix in our list if and only if its score exceeds some threshold
  - 60 for prefixes and 40 for suffixes
  - Threshold is independent of the length of an affix

- since shorter affixes are more likely to be erroneous, we hypothesize that employing larger thresholds for shorter affixes may yield better performance
  - If affix length < 2, multiply threshold by (4 – affix length)
  - E.g., for candidate suffix “j” to remain in the list, it has to attain a score of at least 40*(4-1) = 120.

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction
- Three extensions to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

Detecting Composite Suffixes

- Composite suffix
  - formed by concatenating two or more suffixes (e.g., “ers” = “er” + “s”).

Detecting Composite Suffixes

- Composite suffix
  - formed by concatenating two or more suffixes (e.g., “ers” = “er” + “s”).
- Many suffixes in the induced suffix list are composite

Detecting Composite Suffixes

- Composite suffix
  - formed by concatenating two or more suffixes (e.g., “ers” = “er” + “s”).
- Many suffixes in the induced suffix list are composite
- Need to remove composite suffixes from the list, because their presence could lead to under-segmentation.
  - E.g., “singers” should be segmented as “sing+er+s”. Without composite suffix detection, it will be segmented as “sing+ers”

Detecting Composite Suffixes is not Trivial

- Simple concatenation of two suffixes does not always produce a composite suffix.

Detecting Composite Suffixes is not Trivial

- Simple concatenation of two suffixes does not always produce a composite suffix.

Detecting Composite Suffixes is not Trivial

- Simple concatenation of two suffixes does not always produce a composite suffix.
  - E.g., “en” and “t” are valid suffixes, but “ent” is not a composite suffix

Detecting Composite Suffixes is not Trivial

- Simple concatenation of two suffixes does not always produce a composite suffix.
  - E.g., “en” and “t” are valid suffixes, but “ent” is not a composite suffix

- Same is true for Bangla
  - “TE” ≠ “T” + “E”
  - “Er” ≠ “E” + “r”
  - “Tr” ≠ “T” + “r”

How to Detect Composite Suffixes?

- Employ two criteria
  - Suffix strength
  - Word-level similarity

Suffix Strength

- Observation:

  Let C and S be two suffixes.

  If CS is a composite suffix formed from C and S then

  affix freq (CS) < affix freq (C)

  affix freq (CS) < affix freq (S)

Suffix Strength

Affix frequency: Number of distinct words to which an affix attaches

- Observation:

  Let C and S be two suffixes.

  If CS is a composite suffix formed from C and S then

  affix freq (CS) < affix freq (C)

  affix freq (CS) < affix freq (S)

Suffix Strength

Affix frequency: Number of distinct words to which an affix attaches

- Observation:

  Let C and S be two suffixes.

  If CS is a composite suffix formed from C and S then

  affix freq (CS) < affix freq (C)

  affix freq (CS) < affix freq (S)

  E.g., “ments” is a composite suffix composed of “ment” + “s”

  If we count the affix freqs of “ments”, “ment”, “s” in a large corpus,

  affix freq (“ments”) < affix freq (“ment”)

  affix freq (“ments”) < affix freq (“s”)

Suffix Strength

- Suffix strength alone can be used to determine that a suffix is non-composite

  Consider the Bangla suffix “Er”.

  affix freq (“Er”) = 9817

  affix freq (“E”) = 6218

  affix freq (“r”) = 1247

  So, “Er” can’t be a composite suffix.

Suffix Strength

- Suffix strength alone can be used to determine that a suffix is non-composite

  Consider the Bangla suffix “Er”.

  affix freq (“Er”) = 9817

  affix freq (“E”) = 6218

  affix freq (“r”) = 1247

  So, “Er” can’t be a composite suffix.

- But suffix strength alone is not sufficient for determining that a suffix is composite.

  Consider the Bangla suffix “Ar”

  affix freq (“Ar”) < affix freq (“A”) and affix freq (“r”)

  But, “Ar” is not a composite suffix.

Suffix Strength

- Suffix strength alone can be used to determine that a suffix is non-composite

  Consider the Bangla suffix “Er”.

  affix freq (“Er”) = 9817

  affix freq (“E”) = 6218

  affix freq (“r”) = 1247

  So, “Er” can’t be a composite suffix.

- But suffix strength alone is not sufficient for determining that a suffix is composite.

  Consider the Bangla suffix “Ar”

  affix freq (“Ar”) < affix freq (“A”) and affix freq (“r”)

  But, “Ar” is not a composite suffix.

Need a second condition

Word-Level Similarity

- Observation
  - If a composite suffix (say “ers”) attaches to a word (say “sing”), then most likely its first component suffix (“er”) does.

Word-Level Similarity

- Observation
  - If a composite suffix (say “ers”) attaches to a word (say “sing”), then most likely its first component suffix (“er”) does.
    - A composite suffix and its first component suffix should be similar in terms of the words to which they attach

Word-Level Similarity

- Observation
  - If a composite suffix (say "ers") attaches to a word (say "sing"), then most likely its first component suffix ("er") does.
    - A composite suffix and its first component suffix should be similar in terms of the words to which they attach
  - Formally, if AB is a composite suffix formed from A and B, then

$$\frac{\text{Number of words to which both AB and A attach}}{\text{Number of words to which AB attaches}}$$

should be large

Word-Level Similarity

- Observation
  - If a composite suffix (say "ers") attaches to a word (say "sing"), then most likely its first component suffix ("er") does.
    - A composite suffix and its first component suffix should be similar in terms of the words to which they attach
  - Formally, if AB is a composite suffix formed from A and B, then

$$\boxed{\frac{\text{Number of words to which both AB and A attach}}{\text{Number of words to which AB attaches}}}$$

should be large

Word-level similarity

Our Composite Suffix Detection Algorithm

- Combines these two conditions to determine whether a suffix is composite

- We posit suffix AB as composite if and only if
  1. the suffix strength condition is not violated:
     affix freq(AB) < affix freq(A) and affix freq(AB) < affix freq(B)
  2. the word-level similarity between A and AB is sufficiently high (> 0.6)

The Morpheme Induction Algorithm

- Basic morpheme induction method
  - Prefix and suffix induction
  - Root induction
- Three extensions to the basic induction method
  - Employing length-dependent thresholds
  - Detecting composite suffixes
  - Detecting incorrect attachments

A Motivating Example

- How should “candidate” be segmented?

A Motivating Example

- How should “candidate” be segmented?
  - “candidate” is a root word, so it should **not** be segmented

A Motivating Example

- How should “candidate” be segmented?
  - “candidate” is a root word, so it should **not** be segmented
  - However, given our affix induction algorithm, we would erroneously segment it as “candid” + “ate”

A Motivating Example

- How should “candidate” be segmented?
  - “candidate” is a root word, so it should **not** be segmented
  - However, given our affix induction algorithm, we would erroneously segment it as $\underbrace{\text{“candid”}}_{\textbf{a word in V}} + \underbrace{\text{“ate”}}_{\textbf{an induced suffix}}$

A Motivating Example

- How should “candidate” be segmented?
  - “candidate” is a root word, so it should **not** be segmented
  - However, given our affix induction algorithm, we would erroneously segment it as “candid” + “ate”

a word in V **an induced suffix**

- Problem
  - Failure to recognize that “candidate” is a root itself, resulting in over-segmentation

A Motivating Example

- How should “candidate” be segmented?
  - “candidate” is a root word, so it should **not** be segmented
  - However, given our affix induction algorithm, we would erroneously segment it as “candid” + “ate”

a word in V **an induced suffix**

- Problem
  - Failure to recognize that “candidate” is a root itself, resulting in over-segmentation
- Goal
  - To automatically detect that the attachment of the affix “ate” to “candid” to form “candidate” is incorrect

The Incorrect Attachment Detection Problem

- ”affectionate” = “affection” + “ate” correct
- “candidate” = “candid” + “ate” incorrect

How to Detect Incorrect Attachments?

- A simple algorithm
- Hypothesis

$$w=p+r \text{ or } w=r+s \Rightarrow freq(w) < freq(r)$$

where freq(x) is the corpus frequency of word x

How to Detect Incorrect Attachments?

- A simple algorithm
- Hypothesis

  $w=p+r$ or $w=r+s \Rightarrow freq(w) < freq(r)$

  where freq(x) is the corpus frequency of word x
- It means that the inflectional or derivational form of a root word occurs less frequently than the root word itself

How to Detect Incorrect Attachments?

- A simple algorithm
- Hypothesis

  w=p+r or w=r+s $\Rightarrow$ freq(w) < freq(r)

  where freq(x) is the corpus frequency of word x
- It means that the inflectional or derivational form of a root word occurs less frequently than the root word itself
- Some examples
  - "reopen" = "re" + "open" $\Rightarrow$ freq(reopen) < freq(open)
  - "opening" = "open" + "ing" $\Rightarrow$ freq(opening) < freq(open)
  - "unhealthy" = "un" + "healthy" $\Rightarrow$ freq(unhealthy) < freq(healthy)

To what extent does our hypothesis hold true?

$$\boxed{w=p+r \text{ or } w=r+s \Rightarrow \text{freq}(w) < \text{freq}(r)}$$

To what extent does our hypothesis hold true?

$$\boxed{w=p+r \text{ or } w=r+s \Rightarrow freq(w) < freq(r)}$$

- When evaluated on 286 words randomly chosen from V, the hypothesis is true in 83.56% of the cases.

Applying the Hypothesis to Detect Incorrect Attachments

- Hypothesis: $\boxed{w=p+r \text{ or } w=r+s \Rightarrow \text{freq}(w) < \text{freq}(r)}$

Applying the Hypothesis to Detect Incorrect Attachments

- Hypothesis: $\boxed{w=p+r \text{ or } w=r+s \Rightarrow \text{freq}(w) < \text{freq}(r)}$
- Equivalently, $\boxed{\text{freq}(w) > \text{freq}(r) \Rightarrow w \neq p+r \text{ or } w \neq r+s}$

Applying the Hypothesis to Detect Incorrect Attachments

- Hypothesis: $\boxed{w=p+r \text{ or } w=r+s \Rightarrow freq(w) < freq(r)}$
- Equivalently, $\boxed{freq(w) > freq(r) \Rightarrow w \neq p+r \text{ or } w \neq r+s}$
- Problem
  - since hypothesis is only true in 83.56% of the cases, it would identify many correct attachments as incorrect

Applying the Hypothesis to Detect Incorrect Attachments

- Hypothesis: $\boxed{w=p+r \text{ or } w=r+s \Rightarrow freq(w) < freq(r)}$
- Equivalently, $\boxed{freq(w) > freq(r) \Rightarrow w \neq p+r \text{ or } w \neq r+s}$
- Problem
  - since hypothesis is only true in 83.56% of the cases, it would identify many correct attachments as incorrect
- Solution: relax the hypothesis

$$\boxed{freq(w) > c * freq(r) \Rightarrow w \neq p+r \text{ or } w \neq r+s}$$

  - c=4 for prefixal attachments and 15 for suffixal attachments

Our Unsupervised Word Segmentation Algorithm

1. Morpheme induction
   - Induce morphemes from a vocabulary V (a list of words taken from a large, unannotated corpus)

2. Segmentation
   - Segment a word based on the induced morphemes

Segmentation

- Algorithm adopts a generate-and-remove strategy.

- Given a word to be segmented
  1. Generate all possible segmentations of the word
  2. Apply a sequence of tests to remove candidate segmentations until only one candidate remains

Test 1

- Remove any candidate segmentations $m_1 m_2 \ldots m_n$ that violate any of the following linguistic constraints
  - At least one of m_1, m_2, …, m_n is a root
  - If m_i is a prefix, them m_{i+1} must be a root or a prefix
  - If m_i is a suffix, then m_{i-1} must be a root or a suffix

Test 2

- Retain only those candidate segmentations that have the smallest number of morphemes.

Test 3

- Score each of the remaining candidate segmentations by summing up the score of each morpheme, where
  - The score of a prefix/suffix is its affix frequency, multiplied by the length of the affix
  - The score of a root is the number of morphemes that attach to it, multiplied by the length of the root
- Select the highest-scoring candidate to be the final segmentation

Evaluation

Experimental Setup: Vocabulary Creation

1. Extract vocabulary from a corpus that contains one year of news articles taken from Prothom Alo
2. Pre-process each article by tokenizing it, removing punctuations and other unwanted character sequences

- ~143k distinct words in resulting vocabulary

Experimental Setup: Test Set Preparation

1. Randomly choose 3000 words from V that are at least 3-character long
2. Manually remove proper nouns and words with spelling mistakes
3. Ask two native speakers of Bengali to label the test cases
4. Remove those test cases for which the two annotators produce non-identical segmentations

- 2511 words in resulting test set

Experimental Setup: Evaluation Metrics

- Exact accuracy
  - Percentage of test cases whose proposed segmentation is identical to the correct segmentation

- F-score
  - Harmonic mean of recall and precision

$$\textbf{Recall} = \frac{\text{Number of correctly placed boundaries}}{\text{Number of true morpheme boundaries}}$$

$$\textbf{Precision} = \frac{\text{Number of correctly placed boundaries}}{\text{Number of proposed morpheme boundaries}}$$

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Results

System Variation	Exact Accuracy	Precision	Recall	F-score
Baseline (Linguistica)	37.08	58.25	65.15	61.48
Basic induction	46.67	76.66	66.20	71.04
Composite suffix detection	55.99	79.07	80.61	79.83
Length dependent thresholds	58.38	81.97	79.75	80.85
Incorrect attachment detection	65.83	89.10	80.22	84.43

Conclusions

- A new unsupervised algorithm for Bangla word segmentation
  - Outperforms Linguistica when evaluated on 2511 hand-segmented words
  - Composite suffix detection and incorrect attachment detection contribute significantly to overall performance