Morphological analysis for Russian learner language

Markus Dickinson & Joshua Herring

Dept. of Linguistics, Indiana University

Workshop on Automatic Analysis of Learner Language (AALL-09)
Tempe, AZ; March 10, 2009

Introduction & Motivation

Intelligent computer-aided language learning (ICALL) systems are ideal for language pedagogy

- Intelligent feedback aids awareness of language forms & rules (see Amaral and Meurers 2006)

Introduction & Motivation

Intelligent computer-aided language learning (ICALL) systems are ideal for language pedagogy

- Intelligent feedback aids awareness of language forms & rules (see Amaral and Meurers 2006)

Q: How can we support the provision of intelligent feedback for morphological errors?

Introduction & Motivation

Intelligent computer-aided language learning (ICALL) systems are ideal for language pedagogy

- Intelligent feedback aids awareness of language forms & rules (see Amaral and Meurers 2006)

Q: How can we support the provision of intelligent feedback for morphological errors?

- Should not need to anticipate errors (e.g., Schneider and McCoy 1998)
  - Morphological processing is generally less complex than syntax (e.g., Roark and Sproat 2007)

Introduction & Motivation

Intelligent computer-aided language learning (ICALL) systems are ideal for language pedagogy

- Intelligent feedback aids awareness of language forms & rules (see Amaral and Meurers 2006)

Q: How can we support the provision of intelligent feedback for morphological errors?

- Should not need to anticipate errors (e.g., Schneider and McCoy 1998)
  - Morphological processing is generally less complex than syntax (e.g., Roark and Sproat 2007)

We will outline a morphological error detection & diagnosis procedure for Russian

Overview of talk

What we want to cover today:

1. Define what type of resource(s)/tool(s) we need to analyze learner errors
   - We need to outline the type of errors to be detected
   - We will find that, most importantly, we need an appropriately-structured lexicon

Overview of talk

What we want to cover today:

1. Define what type of resource(s)/tool(s) we need to analyze learner errors
   - We need to outline the type of errors to be detected
   - We will find that, most importantly, we need an appropriately-structured lexicon
2. Acquire an appropriate lexicon
   - We will discuss how to do this quickly

Overview of talk

What we want to cover today:

1. Define what type of resource(s)/tool(s) we need to analyze learner errors
   - We need to outline the type of errors to be detected
   - We will find that, most importantly, we need an appropriately-structured lexicon
2. Acquire an appropriate lexicon
   - We will discuss how to do this quickly
3. Build & evaluate an analyzer using this lexicon

Our particular context

First, a brief note on *why* we are developing a Russian morphological analyzer

Our particular context

First, a brief note on *why* we are developing a Russian morphological analyzer

- We are developing an online workbook for Russian at Indiana University
  - Survival Russian
  - Specialized Russian: Health Care

Our particular context

First, a brief note on *why* we are developing a Russian morphological analyzer

- We are developing an online workbook for Russian at Indiana University
  - Survival Russian
  - Specialized Russian: Health Care
- Currently, the system is essentially a CALL system
  - A morphological analyzer will help provide intelligent feedback on a range of exercises

Our particular context

First, a brief note on *why* we are developing a Russian morphological analyzer

- We are developing an online workbook for Russian at Indiana University
  - Survival Russian
  - Specialized Russian: Health Care
- Currently, the system is essentially a CALL system
  - A morphological analyzer will help provide intelligent feedback on a range of exercises

For more info: come to our talk Saturday morning (3/14) at 8am (Coor L1-20)

Expected error types

Starting point: a taxonomy of expected error types (Dickinson and Herring 2008)

1. Inappropriate stem
   a. Spelling error: Always inappropriate
   b. Semantic/activity error: Inappropriate for this context

Expected error types

Starting point: a taxonomy of expected error types (Dickinson and Herring 2008)

1. Inappropriate stem
   a. Spelling error: Always inappropriate
   b. Semantic/activity error: Inappropriate for this context
2. Inappropriate affix
   a. Spelling error: Always inappropriate
   b. Morphology error: Always inappropriate for, e.g., verbs
   c. Paradigm error: Inappropriate for this word

Expected error types

Starting point: a taxonomy of expected error types (Dickinson and Herring 2008)

1. Inappropriate stem
   a. Spelling error: Always inappropriate
   b. Semantic/activity error: Inappropriate for this context
2. Inappropriate affix
   a. Spelling error: Always inappropriate
   b. Morphology error: Always inappropriate for, e.g., verbs
   c. Paradigm error: Inappropriate for this word
3. Formation error: Inappropriate stem & affix combination

Expected error types

Starting point: a taxonomy of expected error types (Dickinson and Herring 2008)

1. Inappropriate stem
   a. Spelling error: Always inappropriate
   b. Semantic/activity error: Inappropriate for this context
2. Inappropriate affix
   a. Spelling error: Always inappropriate
   b. Morphology error: Always inappropriate for, e.g., verbs
   c. Paradigm error: Inappropriate for this word
3. Formation error: Inappropriate stem & affix combination

We will focus on suffixes, as they encode inflectional morphology in Russian

Inappropriate suffixes

(1) a. начина-ет
nachina-et
begin-3s

b. *начина-еп (#2a)
nachina-ep
begin-?? (invalid suffix of any kind)

c. *начина-ев (#2b)
nachina-ev
begin-?? (masc.gen.pl *noun* affix)

d. *начина-ит (#2c)
nachina-it
begin-3S (different verb paradigm)

Formation errors (#3)

Some verbs change stem form, depending on suffix vowel:

(2) a. мог-ут
mog-ut
can-3p

b. мож-ет
mozh-et
can-3s

c. *мож-ут (#3)
mozh-ut
can-3p (wrong formation)

Multiple analyses

(3) *мож-ут
mozh-ut
can-3p

At least two possible analyses:

- Formation error (#3): Learner attempting to form third person plural (*mog-ut*)

Multiple analyses

(3) *мож-ут
mozh-ut
can-3p

At least two possible analyses:

- Formation error (#3): Learner attempting to form third person plural (*mog-ut*)
- Spelling error (#2a): Learner attempting to form third person singular (*mozh-et*)

Multiple analyses

(3) *мож-ут
mozh-ut
can-3p

At least two possible analyses:

- Formation error (#3): Learner attempting to form third person plural (*mog-ut*)
- Spelling error (#2a): Learner attempting to form third person singular (*mozh-et*)

⇒ We need multiple analyses until we have more information (cf. also Dickinson and Herring 2008)

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

0. Correct: the stem and suffix occur in the lexicon together

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

0. Correct: the stem and suffix occur in the lexicon together
1. a. Stem spelling error [later]
   b. Activity error [later]
2. a. Suffix spelling error [later]

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

0. Correct: the stem and suffix occur in the lexicon together
1. a. Stem spelling error [later]
   b. Activity error [later]
2. a. Suffix spelling error [later]
   b. Morphology error: stem & suffix have incompatible tags
      - e.g., N vs. V

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

0. Correct: the stem and suffix occur in the lexicon together
1. a. Stem spelling error [later]
   b. Activity error [later]
2. a. Suffix spelling error [later]
   b. Morphology error: stem & suffix have incompatible tags
      - e.g., N vs. V
   c. Paradigm error: the stem has a different suffix in the lexicon with the same tag
      - e.g., *-et* instead of *-it* (but both `Vmip3s-a-p`)

Detecting & classifying learner errors

Q: How can we detect & classify these types of errors?

- A: See *how* a stem & suffix do/don't match

0. Correct: the stem and suffix occur in the lexicon together
1. a. Stem spelling error [later]
   b. Activity error [later]
2. a. Suffix spelling error [later]
   b. Morphology error: stem & suffix have incompatible tags
      - e.g., N vs. V
   c. Paradigm error: the stem has a different suffix in the lexicon with the same tag
      - e.g., *-et* instead of *-it* (but both `Vmip3s-a-p`)
3. Formation error: stem & suffix are compatible, but stem has no such suffix tag in lexicon
   - e.g., *mozh* has no `Vmip3s-a-p` suffix

Making inferences

Paradigm errors (#2c)

(4) *начина-ит
nachina-it
begin+Vmip3s-a-p (wrong verb paradigm)

Stem & suffix do not occur together in the lexicon

Making inferences

Paradigm errors (#2c)

(4) *начина-ит
nachina-it
begin+`Vmip3s-a-p` (wrong verb paradigm)

Stem & suffix do not occur together in the lexicon

- *-it* has certain morphosyntactic properties: `Vmip3s-a-p`

Making inferences

Paradigm errors (#2c)

(4) *начина-ит
nachina-it
begin+`Vmip3s-a-p` (wrong verb paradigm)

Stem & suffix do not occur together in the lexicon

- *-it* has certain morphosyntactic properties: `Vmip3s-a-p`
- There is a variant (*-et*) with same properties
  - Variant is in the lexicon with this stem

(5) начина-ет
nachina-et
begin+`Vmip3s-a-p`

Making inferences

Formation errors (#3)

(6) *мож-ут
mozh-ut
can+Vmip3p-a-p (wrong formation)

Suffix tag is compatible with stem

- Suffix tag never observed with this stem
  - Not just the literal suffix, but its morphosyntactic properties have not been seen with this stem

Making inferences

Formation errors (#3)

(6) *мож-ут
mozh-ut
can+Vmip3p-a-p (wrong formation)

Suffix tag is compatible with stem

- Suffix tag never observed with this stem
  - Not just the literal suffix, but its morphosyntactic properties have not been seen with this stem
- *If the lexicon is complete*, we can infer that there is no such suffix tag for this stem

Making inferences

Formation errors (#3)

(6) *мож-ут
mozh-ut
can+Vmip3p-a-p (wrong formation)

Suffix tag is compatible with stem

- Suffix tag never observed with this stem
  - Not just the literal suffix, but its morphosyntactic properties have not been seen with this stem
- *If the lexicon is complete*, we can infer that there is no such suffix tag for this stem
  - One way to combat lexicon incompleteness: Get as big a lexicon as possible

Desired lexical entries

From all this, we want to get the following for each word:

- stem
- stem tag
- suffix
- suffix tag

Desired lexical entries

From all this, we want to get the following for each word:

- stem
- stem tag
- suffix
- suffix tag

e.g., possible lexical entries for *mog-* verbs:

(7) a. мог, `Vm-----a-p`, у, `Vmip1s-a-p`

b. мож, `Vmip---a-p`, ет, `Vmip3s-a-p`

c. мог, `Vm-----a-p`, `NULL`, `Vmis-sma-p`

Desired lexical entries

From all this, we want to get the following for each word:

- stem
- stem tag
- suffix
- suffix tag

e.g., possible lexical entries for *mog-* verbs:

(7) a. мог, `Vm-----a-p`, у, `Vmip1s-a-p`

b. мож, `Vmip---a-p`, ет, `Vmip3s-a-p`

c. мог, `Vm-----a-p`, `NULL`, `Vmis-sma-p`

NB: multiple suffixes are combined into a single form

- Should be okay, since each POS tag encodes the properties of all suffixes in a word

Enriching a POS lexicon

Why not re-use a Russian morphological analyzer?

- They only return correct analyses (e.g., Gelbukh and Sidorov 2003; Segalovich 2003; Yablonsky 1999)

Enriching a POS lexicon

Why not re-use a Russian morphological analyzer?

- They only return correct analyses (e.g., Gelbukh and Sidorov 2003; Segalovich 2003; Yablonsky 1999)

Freely-available POS lexicon (Sharoff et al. 2008)

- 244,751 unique tokens, with all possible POS tags and frequency counts of each tag
  - POS tags are bundles of morphological information

Enriching a POS lexicon

Why not re-use a Russian morphological analyzer?

- They only return correct analyses (e.g., Gelbukh and Sidorov 2003; Segalovich 2003; Yablonsky 1999)

Freely-available POS lexicon (Sharoff et al. 2008)

- 244,751 unique tokens, with all possible POS tags and frequency counts of each tag
  - POS tags are bundles of morphological information
- We just need to determine morphemes & boundaries from full words

Enriching a POS lexicon

Why not re-use a Russian morphological analyzer?

- They only return correct analyses (e.g., Gelbukh and Sidorov 2003; Segalovich 2003; Yablonsky 1999)

Freely-available POS lexicon (Sharoff et al. 2008)

- 244,751 unique tokens, with all possible POS tags and frequency counts of each tag
  - POS tags are bundles of morphological information
- We just need to determine morphemes & boundaries from full words
  - Saves time in writing desired entries
    - cf. 5 years to build a lexicon of German (Geyken and Hanneforth 2005)

Segment finding

Developed a simple algorithm to segment words into morphemes

Core idea: the same feature specifications indicate similarity of morphemes (cf., e.g., Ćavar et al. 2008)

- Bears similarity to affix positing in Schone and Jurafsky (2001) and Gaussier (1999)

Segment finding algorithm

1. Group all analyses (word, POS pairs) with same POS tag

Segment finding algorithm

1. Group all analyses (word, POS pairs) with same POS tag
2. For each POS tag, determine set of **possible suffixes**
   - Find longest common suffix (possibly `NULL`) of 2 words

Segment finding algorithm

1. Group all analyses (word, POS pairs) with same POS tag
2. For each POS tag, determine set of **possible suffixes**
   - Find longest common suffix (possibly `NULL`) of 2 words
3. Filter out potentially illegimate suffixes
   - Legitimacy test based on the idea that real suffixes will accidentally lead to longer "suffixes"

Segment finding algorithm

1. Group all analyses (word, POS pairs) with same POS tag
2. For each POS tag, determine set of **possible suffixes**
   - Find longest common suffix (possibly `NULL`) of 2 words
3. Filter out potentially illegimate suffixes
   - Legitimacy test based on the idea that real suffixes will accidentally lead to longer "suffixes"
4. With set of possible suffixes (and tags), find each word's **possible stem** based on the *most likely suffix*
   - Basic heuristic: most frequent matching suffix (not including `NULL`)

Segment finding algorithm

1. Group all analyses (word, POS pairs) with same POS tag
2. For each POS tag, determine set of **possible suffixes**
   - Find longest common suffix (possibly `NULL`) of 2 words
3. Filter out potentially illegimate suffixes
   - Legitimacy test based on the idea that real suffixes will accidentally lead to longer "suffixes"
4. With set of possible suffixes (and tags), find each word's **possible stem** based on the *most likely suffix*
   - Basic heuristic: most frequent matching suffix (not including `NULL`)
5. For each stem and suffix combination (i.e., segmented word), hypothesize a **stem tag**
   - Find commonality of all tags a stem can have
   - Allows us to determine compatible endings

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs
   - Can restrict suffix to a certain size

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs
   - Can restrict suffix to a certain size
   - Can easily restrict to match activity constraints (#1b)

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs
   - Can restrict suffix to a certain size
   - Can easily restrict to match activity constraints (#1b)
2. Look up each stem and suffix in lexicon

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs
   - Can restrict suffix to a certain size
   - Can easily restrict to match activity constraints (#1b)
2. Look up each stem and suffix in lexicon
   - Potentially check repairs (insertions, deletions, substitutions) on either stem or suffix (#1a, #2a)

Analysis

Now have each word's stem, stem tag, suffix, & suffix tag

- Next step: put the lexicon to work analyzing input words

Goal: outline the appropriateness of using such a morphosyntactic lexicon for analyzing learner language

1. Divide word into all possible stem & suffix pairs
   - Can restrict suffix to a certain size
   - Can easily restrict to match activity constraints (#1b)
2. Look up each stem and suffix in lexicon
   - Potentially check repairs (insertions, deletions, substitutions) on either stem or suffix (#1a, #2a)
3. Compare results of each stem & suffix analysis, to get error information

Evaluation

Three questions we want to address, directly or indirectly:

Evaluation

Three questions we want to address, directly or indirectly:

1. Are the assigned tags doing any linguistic work?
   - Do they capture real generalizations over the language that learners need to acquire?

Evaluation

Three questions we want to address, directly or indirectly:

1. Are the assigned tags doing any linguistic work?
   - Do they capture real generalizations over the language that learners need to acquire?
2. Are the correct tags for a word being appropriately generated?

Evaluation

Three questions we want to address, directly or indirectly:

1. Are the assigned tags doing any linguistic work?
   - Do they capture real generalizations over the language that learners need to acquire?
2. Are the correct tags for a word being appropriately generated?
3. How much are we overgenerating analyses, and how can we appropriately reduce the overgeneration?

The data

Data split from our lexicon:

- Training data: 90% of the words (211,716)
- Known testing data: 10%, overlapping with training
- Unknown testing data: 10% non-overlapping

The data

Data split from our lexicon:

- Training data: 90% of the words (211,716)
- Known testing data: 10%, overlapping with training
- Unknown testing data: 10% non-overlapping

In lieu of real learner data, we corrupt known testing data:

- every word has one one randomly-deleted, randomly-inserted or randomly-substituted character

The data

Data split from our lexicon:

- Training data: 90% of the words (211,716)
- Known testing data: 10%, overlapping with training
- Unknown testing data: 10% non-overlapping

In lieu of real learner data, we corrupt known testing data:

- every word has one one randomly-deleted, randomly-inserted or randomly-substituted character

We report:

- number of analyses for each error type, on average
- *recall*: percentage of correct analyses returned by system

Initial results

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.25	0.43	0.65	46.51	100.0%
	4	1.22	0.43	0.65	46.49	98.5%
Unkn.	n/a	0	0.33	0.23	34.49	83.9%
	4	0	0.33	0.23	34.42	81.9%
Sub.	4	0	0.05	0.02	2.94	3.3%
Del.	4	0	0.38	0.28	27.13	22.1%
Ins.	4	0	0.01	0.00	0.46	0.8%

Initial results

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.25	0.43	0.65	46.51	100.0%
	4	1.22	0.43	0.65	46.49	98.5%
Unkn.	n/a	0	0.33	0.23	34.49	83.9%
	4	0	0.33	0.23	34.42	81.9%
Sub.	4	0	0.05	0.02	2.94	3.3%
Del.	4	0	0.38	0.28	27.13	22.1%
Ins.	4	0	0.01	0.00	0.46	0.8%

- Large number of #2b analyses (morphology error)
  - Known words: #2b adds almost no new correct analyses

Initial results

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.25	0.43	0.65	46.51	100.0%
	4	1.22	0.43	0.65	46.49	98.5%
Unkn.	n/a	0	0.33	0.23	34.49	83.9%
	4	0	0.33	0.23	34.42	81.9%
Sub.	4	0	0.05	0.02	2.94	3.3%
Del.	4	0	0.38	0.28	27.13	22.1%
Ins.	4	0	0.01	0.00	0.46	0.8%

- Large number of #2b analyses (morphology error)
  - Known words: #2b adds almost no new correct analyses
  - Unknown words: #2b accounts for high recall (otherwise: 1.5%)
    - system using suffix to guess category

Initial results

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.25	0.43	0.65	46.51	100.0%
	4	1.22	0.43	0.65	46.49	98.5%
Unkn.	n/a	0	0.33	0.23	34.49	83.9%
	4	0	0.33	0.23	34.42	81.9%
Sub.	4	0	0.05	0.02	2.94	3.3%
Del.	4	0	0.38	0.28	27.13	22.1%
Ins.	4	0	0.01	0.00	0.46	0.8%

- Large number of #2b analyses (morphology error)
  - Known words: #2b adds almost no new correct analyses
  - Unknown words: #2b accounts for high recall (otherwise: 1.5%)
    - system using suffix to guess category
- Words needing repair have different patterns
  - Encouraging: correct analysis should involve repair

Comparison to naive method

Compare to randomized segment finding (suffix ≤ 7):

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.52	1.68	1.11	161.46	100.0%
	4	1.16	1.49	1.08	159.85	97.3%
Unkn.	n/a	0	1.16	0.38	64.63	94.3%
	4	0	0.98	0.36	62.77	89.7%
Sub.	4	0	0.34	0.06	6.53	15.7%
Del.	4	0	1.89	0.55	41.66	53.4%
Ins.	4	0	0.11	0.02	1.74	11.7%

Comparison to naive method

Compare to randomized segment finding (suffix ≤ 7):

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.52	1.68	1.11	161.46	100.0%
	4	1.16	1.49	1.08	159.85	97.3%
Unkn.	n/a	0	1.16	0.38	64.63	94.3%
	4	0	0.98	0.36	62.77	89.7%
Sub.	4	0	0.34	0.06	6.53	15.7%
Del.	4	0	1.89	0.55	41.66	53.4%
Ins.	4	0	0.11	0.02	1.74	11.7%

- High recall for unknown words: lots of suffixes to use

Comparison to naive method

Compare to randomized segment finding (suffix ≤ 7):

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.52	1.68	1.11	161.46	100.0%
	4	1.16	1.49	1.08	159.85	97.3%
Unkn.	n/a	0	1.16	0.38	64.63	94.3%
	4	0	0.98	0.36	62.77	89.7%
Sub.	4	0	0.34	0.06	6.53	15.7%
Del.	4	0	1.89	0.55	41.66	53.4%
Ins.	4	0	0.11	0.02	1.74	11.7%

- High recall for unknown words: lots of suffixes to use
  - Our algorithm: 285 distinct suffix forms corresponding to 1510 total analyses (i.e., suffix-tag pairings)

Comparison to naive method

Compare to randomized segment finding (suffix ≤ 7):

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.52	1.68	1.11	161.46	100.0%
	4	1.16	1.49	1.08	159.85	97.3%
Unkn.	n/a	0	1.16	0.38	64.63	94.3%
	4	0	0.98	0.36	62.77	89.7%
Sub.	4	0	0.34	0.06	6.53	15.7%
Del.	4	0	1.89	0.55	41.66	53.4%
Ins.	4	0	0.11	0.02	1.74	11.7%

- High recall for unknown words: lots of suffixes to use
  - Our algorithm: 285 distinct suffix forms corresponding to 1510 total analyses (i.e., suffix-tag pairings)
  - Random splits: 37,733 suffixes for 59,860 analyses

Comparison to naive method

Compare to randomized segment finding (suffix ≤ 7):

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	n/a	1.52	1.68	1.11	161.46	100.0%
	4	1.16	1.49	1.08	159.85	97.3%
Unkn.	n/a	0	1.16	0.38	64.63	94.3%
	4	0	0.98	0.36	62.77	89.7%
Sub.	4	0	0.34	0.06	6.53	15.7%
Del.	4	0	1.89	0.55	41.66	53.4%
Ins.	4	0	0.11	0.02	1.74	11.7%

- High recall for unknown words: lots of suffixes to use
  - Our algorithm: 285 distinct suffix forms corresponding to 1510 total analyses (i.e., suffix-tag pairings)
  - Random splits: 37,733 suffixes for 59,860 analyses

High amount of compression on the number of suffixes and analyses suggests linguistic generalizations

Results with repairs

Spelling errors (#1a/#2a) bring additional possibilities:

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	4	14.24	19.29	14.34	1407.88	99.0%
Unkn.	4	2.72	10.96	7.36	985.71	94.2%
Sub.	4	1.94	5.19	2.87	312.21	**98.5%**
Del.	4	3.19	15.47	9.59	974.89	**98.7%**
Ins.	4	1.55	1.34	1.11	100.21	**96.7%**

Results with repairs

Spelling errors (#1a/#2a) bring additional possibilities:

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	4	14.24	19.29	14.34	1407.88	99.0%
Unkn.	4	2.72	10.96	7.36	985.71	94.2%
Sub.	4	1.94	5.19	2.87	312.21	**98.5%**
Del.	4	3.19	15.47	9.59	974.89	**98.7%**
Ins.	4	1.55	1.34	1.11	100.21	**96.7%**

- Error case #2b is extremely noisy

Results with repairs

Spelling errors (#1a/#2a) bring additional possibilities:

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	4	14.24	19.29	14.34	1407.88	99.0%
Unkn.	4	2.72	10.96	7.36	985.71	94.2%
Sub.	4	1.94	5.19	2.87	312.21	**98.5%**
Del.	4	3.19	15.47	9.59	974.89	**98.7%**
Ins.	4	1.55	1.34	1.11	100.21	**96.7%**

- Error case #2b is extremely noisy
  - Main reason is that we allow any stem-suffix mismatch to count as a #2b case

Results with repairs

Spelling errors (#1a/#2a) bring additional possibilities:

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	4	14.24	19.29	14.34	1407.88	99.0%
Unkn.	4	2.72	10.96	7.36	985.71	94.2%
Sub.	4	1.94	5.19	2.87	312.21	**98.5%**
Del.	4	3.19	15.47	9.59	974.89	**98.7%**
Ins.	4	1.55	1.34	1.11	100.21	**96.7%**

- Error case #2b is extremely noisy
  - Main reason is that we allow any stem-suffix mismatch to count as a #2b case
  - Restricting this by only allowing certain mismatches could lead to a sensible reduction

Results with repairs

Spelling errors (#1a/#2a) bring additional possibilities:

Data	Suf.	#0	#2c	#3	#2b	Recall
Known	4	14.24	19.29	14.34	1407.88	99.0%
Unkn.	4	2.72	10.96	7.36	985.71	94.2%
Sub.	4	1.94	5.19	2.87	312.21	**98.5%**
Del.	4	3.19	15.47	9.59	974.89	**98.7%**
Ins.	4	1.55	1.34	1.11	100.21	**96.7%**

- Error case #2b is extremely noisy
  - Main reason is that we allow any stem-suffix mismatch to count as a #2b case
  - Restricting this by only allowing certain mismatches could lead to a sensible reduction
- Can also reduce over-generation by considering repairs only when not enough analyses have been generated

Other ways to reduce over-generation

The results on the previous slide are the result of first repairing and then comparing stem & suffix

Other ways to reduce over-generation

The results on the previous slide are the result of first repairing and then comparing stem & suffix

- This means that we actually have two errors for #2c, #3, & #2b on previous slide

Other ways to reduce over-generation

The results on the previous slide are the result of first repairing and then comparing stem & suffix

- This means that we actually have two errors for #2c, #3, & #2b on previous slide
- Sensible heuristic: allow only one error per word

Other ways to reduce over-generation

The results on the previous slide are the result of first repairing and then comparing stem & suffix

- This means that we actually have two errors for #2c, #3, & #2b on previous slide
- Sensible heuristic: allow only one error per word

Additionally, there are more suffixes in the lexicon than learners will know

Other ways to reduce over-generation

The results on the previous slide are the result of first repairing and then comparing stem & suffix

- This means that we actually have two errors for #2c, #3, & #2b on previous slide
- Sensible heuristic: allow only one error per word

Additionally, there are more suffixes in the lexicon than learners will know

- We can trim the lexicon to only include level-appropriate distinctions

Summary & Outlook

Summary:

- Outlined a type of lexicon which is appropriate for providing feedback on potentially ill-formed language
- Built such a lexicon from a freely-available POS lexicon using a handful of sensible heuristics
- Demonstrated the utility of using such a lexicon

Summary & Outlook

Summary:

- Outlined a type of lexicon which is appropriate for providing feedback on potentially ill-formed language
- Built such a lexicon from a freely-available POS lexicon using a handful of sensible heuristics
- Demonstrated the utility of using such a lexicon

Next Steps:

- Clean & augment lexicon by hand:
  - will work quickly, given simplicity of the lexicon
  - will provide test data for segment-finding
- Implement analyzer as a finite-state automata (Ćavar et al. 2008; Geyken and Hanneforth 2005)
- Try on real learner langauge
  - Use real errors to guide the analyzer in its stem-suffix mismatches

Acknowledgments

We would like to thank:

- Anna Feldman & Jirka Hana for advice on Russian resources
- The Indiana University Computational Linguistics discussion group for comments & feedback

This research was supported by grant P116S070001 through the U.S. Department of Education's Fund for the Improvement of Postsecondary Education

References

Amaral, Luiz and Detmar Meurers (2006). Where does ICALL Fit into Foreign Language Teaching? Talk given at CALICO Conference. University of Hawaii, http://purl.org/net/icall/handouts/calico06-amaral-meurers.pdf.

Ćavar, Damir, Ivo-Pavao Jazbec and Siniša Runjaić (2008). Interoperability and Rapid Bootstrapping of Morphological Parsing and Annotation Automata. In *Proceedings of IS-LTC 08*. Ljubljana, Slovenia.

Dickinson, Markus and Joshua Herring (2008). Developing Online ICALL Exercises for Russian. In *The 3rd Workshop on Innovative Use of NLP for Building Educational Applications*. Columbus, OH, pp. 1–9.

Gaussier, Eric (1999). Unsupervised learning of derivational morphology from inflectional lexicons. In *Proceedings of the Workshop on Unsupervised Learning in NLP*. pp. 24–30.

Gelbukh, Alexander and Grigori Sidorov (2003). Approach to construction of automatic morphological analysis systems for inflective languages with little effort. In *Proceedings of CICLing-03*. Springer-Verlag, no. 2588 in Lecture Notes in Computer Science, pp. 215–220.

Geyken, Alexander and Thomas Hanneforth (2005). TAGH: A Complete Morphology for German Based on Weighted Finite State Automata. In *FSMNLP 2005*. Springer, vol. 4002 of *Lecture Notes in Computer Science*, pp. 55–66.

Roark, Brian and Richard Sproat (2007). *Computational Approaches to Morphology and Syntax*. Oxford University Press.

Schneider, David A. and Kathleen F. McCoy (1998). Recognizing Syntactic Errors in the Writing of Second Language Learners. In *Proceedings of ACL-COLING-98*. Montreal, pp. 1198–1204.

Schone, Patrick and Daniel Jurafsky (2001). Knowledge-Free Induction of Inflectional Morphologies. In *Proceedings of NAACL-01*. Pittsburgh, PA.

Segalovich, Ilya (2003). A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. http://company.yandex.ru/articles/iseg-las-vegas.html.

Sharoff, Serge, Mikhail Kopotev, Tomaž Erjavec, Anna Feldman and Dagmar Divjak (2008). Designing and evaluating Russian tagsets. In *Proceedings of LREC-08*. Marrakech.

Yablonsky, Serge (1999). Russian Morphological Analysis. In *Proceedings of VEXTAL'99*. Venezia, San Servolo, pp. 83–90.

Filtering step (3) of segment finding

Consider `Npfpay` proper nouns:

- зар (*zar*)
- тамар (*tamar*)

System wrongly hypothesizes -ap (*-ar*) suffix

Idea: If suffix is legitimate, should be accidental longer "suffixes"

- (*-at'*) is legitimate infinitive suffix
- Many `Vmn----a-p` words with longer common substrings: иг**рать** (*igrat'*, 'to play') & б**рать** (*brat'*, 'to take')

If "suffix" is accident, less likely for accidental longer suffixes

- -ap (*-ar*) for `Npfpay` has no longer suffixes

⇒ Remove proposed suffixes without longer variants for same POS class