Approximate String Search and Matching

COMP90049
COMP30018
Knowledge Technologies

Lea Frermann and Justin Zobel and Karin Verspoor

Semester 2, 2019

THE UNIVERSITY OF MELBOURNE

Roadmap

Last time

Approximate string matching – part 1

- motivation
- methods

This time

Approximate string matching – part 2

- more methods
- text preprocessing
- evaluation
- assignment 1

N-Gram Distance

Intro

- another method for finding best approximate string match
- a true distance

N-Gram Distance

Intro

- another method for finding best approximate string match
- a true distance

What is an n-gram? A (character) substring of length *n*

- 2-grams of `crat`: `cr`, `ra`, `at`; **or**
- 2-grams of `crat`: `#c`, `cr`, `ra`, `at`, `t#` (sometimes)
- 3-grams of `crat`: `##c`, `#cr`, `cra`, `rat`, `at#`, `t##`

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) = \texttt{\#c, cr, ra, at, t\#}$
- 2-grams of `cart`: $G_2(\texttt{cart}) = \texttt{\#c, ca, ar, rt, t\#}$
- 2-grams of `arts`: $G_2(\texttt{arts}) = \texttt{\#a, ar, rt, ts, s\#}$

N-Gram Distance in Action

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) = \texttt{\#c, cr, ra, at, t\#}$
- 2-grams of `cart`: $G_2(\texttt{cart}) = \texttt{\#c, ca, ar, rt, t\#}$
- 2-grams of `arts`: $G_2(\texttt{arts}) = \texttt{\#a, ar, rt, ts, s\#}$

N-Gram Distance between $G_n(s)$ and $G_n(t)$:

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

N-Gram Distance in Action

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) = \texttt{\#c, cr, ra, at, t\#}$
- 2-grams of `cart`: $G_2(\texttt{cart}) = \texttt{\#c, ca, ar, rt, t\#}$
- 2-grams of `arts`: $G_2(\texttt{arts}) = \texttt{\#a, ar, rt, ts, s\#}$

N-Gram Distance between $G_n(s)$ and $G_n(t)$:

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

2-Gram Distance between `crat` and `cart`:

N-Gram Distance in Action

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) =$ `#c, cr, ra, at, t#`
- 2-grams of `cart`: $G_2(\texttt{cart}) =$`#c, ca, ar, rt, t#`
- 2-grams of `arts`: $G_2(\texttt{arts}) =$`#a, ar, rt, ts, s#`

N-Gram Distance between $G_n(s)$ and $G_n(t)$:

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

2-Gram Distance between `crat` and `cart`:

$$|G_2(\texttt{crat})| + |G_2(\texttt{cart})| - 2 \times |G_2(\texttt{crat}) \cap G_2(\texttt{cart})|$$
$$= 5 + 5 - 2 \times 2 = 6$$

N-Gram Distance in Action

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) = \texttt{\#c, cr, ra, at, t\#}$
- 2-grams of `cart`: $G_2(\texttt{cart}) = \texttt{\#c, ca, ar, rt, t\#}$
- 2-grams of `arts`: $G_2(\texttt{arts}) = \texttt{\#a, ar, rt, ts, s\#}$

N-Gram Distance between $G_n(s)$ and $G_n(t)$:

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

2-Gram Distance between `crat` and `cart`:

$$\begin{aligned} & |G_2(\texttt{crat})| + |G_2(\texttt{cart})| - 2 \times |G_2(\texttt{crat}) \cap G_2(\texttt{cart})| \\ = & \; 5 + 5 - 2 \times 2 = \; 6 \end{aligned}$$

2-Gram Distance between `crat` and `arts`:

$$\begin{aligned} & |G_2(\texttt{crat})| + |G_2(\texttt{arts})| - 2 \times |G_2(\texttt{crat}) \cap G_2(\texttt{arts})| \\ = & \; 5 + 5 - 2 \times 0 = \; 10 \end{aligned}$$

N-Gram Distance in Action

N-gram distance example

- 2-grams of `crat`: $G_2(\texttt{crat}) = \texttt{\#c, cr, ra, at, t\#}$
- 2-grams of `cart`: $G_2(\texttt{cart}) = \texttt{\#c, ca, ar, rt, t\#}$
- 2-grams of `arts`: $G_2(\texttt{arts}) = \texttt{\#a, ar, rt, ts, s\#}$

N-Gram Distance between $G_n(s)$ and $G_n(t)$:

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

2-Gram Distance between `crat` and `cart`:

$$\begin{aligned} & |G_2(\texttt{crat})| + |G_2(\texttt{cart})| - 2 \times |G_2(\texttt{crat}) \cap G_2(\texttt{cart})| \\ = {} & 5 + 5 - 2 \times 2 = \; 6 \end{aligned}$$

2-Gram Distance between `crat` and `arts`:

$$\begin{aligned} & |G_2(\texttt{crat})| + |G_2(\texttt{arts})| - 2 \times |G_2(\texttt{crat}) \cap G_2(\texttt{arts})| \\ = {} & 5 + 5 - 2 \times 0 = \; 10 \end{aligned}$$

cart is (again!) the better match!

N-Gram Distance Efficiency

- Occasionally useful as a simpler variant of (Global) Edit Distance

N-Gram Distance Efficiency

- Occasionally useful as a simpler variant of (Global) Edit Distance
- More sensitive to long substring matches, less sensitive to relative ordering of strings (matches can be anywhere!)

N-Gram Distance Efficiency

- Occasionally useful as a simpler variant of (Global) Edit Distance
- More sensitive to long substring matches, less sensitive to relative ordering of strings (matches can be anywhere!)
- Despite its simplicity, takes roughly the same time to compare entire dictionary

N-Gram Distance Efficiency

- Occasionally useful as a simpler variant of (Global) Edit Distance
- More sensitive to long substring matches, less sensitive to relative ordering of strings (matches can be anywhere!)
- Despite its simplicity, takes roughly the same time to compare entire dictionary
- Quite useless for very long strings and/or very small alphabets (Why?)

N-Gram Distance Efficiency

- Occasionally useful as a simpler variant of (Global) Edit Distance
- More sensitive to long substring matches, less sensitive to relative ordering of strings (matches can be anywhere!)
- Despite its simplicity, takes roughly the same time to compare entire dictionary
- Quite useless for very long strings and/or very small alphabets (Why?)
- **Potentially useful for (approximate) prefixes / suffixes**, e.g., Street $\rightarrow$ St; or smog $\rightarrow$ smoke

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a, b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a, b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

- m is the number of matches by greedy alignment (c,r,a,t). Two characters match if their positions in a and b are similar enough

$$m > 0 \text{ iff difference in mention positions} <= \lfloor \frac{max(|a|, |b|)}{2} - 1 \rfloor$$

$$\lfloor \frac{max(|a|, |b|)}{2} - 1 \rfloor = \lfloor \frac{6}{2} - 1 \rfloor = 2$$

cart and **crat**ec $\rightarrow m = 4$!

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a,b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

- m is the number of matches by greedy alignment (c,r,a,t). Two characters match if their positions in a and b are similar enough

  $$m > 0 \text{ iff difference in mention positions } <= \lfloor \frac{max(|a|,|b|)}{2} - 1 \rfloor$$

  $$\lfloor \frac{max(|a|,|b|)}{2} - 1 \rfloor = \lfloor \frac{6}{2} - 1 \rfloor = 2$$

  cart and **crat**ec $\rightarrow m = 4!$

- t is $\frac{1}{2}$ the number of transpositions (i.e., matches which are not in the same position)
  c**ar**t and c**ra**tec $\rightarrow$ $t = \frac{1}{2} \times 2 = 1$

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a, b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a,b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

Jaro-**Winkler extension**: give more weight to strings with matching prefixes

$$sim_w(a,b) = sim_j(a,b) + \ell p(1 - sim_j)$$

- ℓ length of the true common prefix (typically capped at 4)
- p parameter, weight of the prefix match (typically ≈ 0.1)

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a,b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

Jaro-**Winkler extension**: give more weight to strings with matching prefixes

$$sim_w(a,b) = sim_j(a,b) + \ell p(1 - sim_j)$$

- ℓ length of the true common prefix (typically capped at 4)
- p parameter, weight of the prefix match (typically ≈ 0.1)

In pairs, calculate sim_w(cart, cratec)

Jaro-Winkler Similarity

Again, useful for matching (approximate) prefixes / suffixes.

a=cart; b=cratec

$$sim_j(a,b) = \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right)$$

Jaro-**Winkler extension**: give more weight to strings with matching prefixes

$$sim_w(a,b) = sim_j(a,b) + \ell p(1 - sim_j)$$

- ℓ length of the true common prefix (typically capped at 4)
- p parameter, weight of the prefix match (typically ≈ 0.1)

$$sim_j(a,b) = \frac{1}{3}\left(\frac{4}{4} + \frac{4}{6} + \frac{4-1}{4}\right) \approx 0.81$$

$$sim_w(a,b) = 0.81 + 0.1 \times 1 \times (1 - 0.81) \approx 0.83$$

Evaluation: consider whether the system is effective at solving the user's problem

Evaluating an Approximate Matching System

Evaluation: consider whether the system is effective at solving the user's problem

For example:

- for a misspelled word, does the system identify the correct word?

Evaluating an Approximate Matching System

Evaluation: consider whether the system is effective at solving the user's problem

For example:

- for a misspelled word, does the system identify the correct word?
- for a 'lexical blend' our algorithm found, does it correspond to a true word in our dictionary?

Evaluation: consider whether the system is effective at solving the user's problem

For example:

- for a misspelled word, does the system identify the correct word?
- for a 'lexical blend' our algorithm found, does it correspond to a true word in our dictionary?

Evaluating an Approximate Matching System

Evaluation: consider whether the system is effective at solving the user's problem

For example:

- for a misspelled word, does the system identify the correct word?
- for a 'lexical blend' our algorithm found, does it correspond to a true word in our dictionary?

To evaluate, we need:

1. **Predictions** of our algorithm
   (e.g., corrections for misspelled words / lexical blend predictions)
2. A 'Gold standard' as **ground truth**

   (incl., the correctspelling / known lexical blends)
3. An **evaluation metric**

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?
`brunch`	yes	yes

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?
brunch	yes	yes
dinner	yes	–

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?
`brunch`	yes	yes
`dinner`	yes	–
`smog`	yes	yes

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?
`brunch`	yes	yes
`dinner`	yes	–
`smog`	yes	yes
`football`	yes	–

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?
`brunch`	yes	yes
`dinner`	yes	–
`smog`	yes	yes
`football`	yes	–
`brexit`	–	yes
...	...	...

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?	Right/Wrong?
`brunch`	yes	yes	✓
`dinner`	yes	–	×
`smog`	yes	yes	✓
`football`	yes	–	×
`brexit`	–	yes	×
...	...	...	

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?	Right/Wrong?
brunch	yes	yes	✓
dinner	yes	–	×
smog	yes	yes	✓
football	yes	–	×
brexit	–	yes	×
...	...	...	

Accuracy:
fraction of correct responses among total number of words ($\frac{2}{5}$)

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?	Right/Wrong?
`brunch`	yes	yes	✓
`dinner`	yes	–	×
`smog`	yes	yes	✓
`football`	yes	–	×
`brexit`	–	yes	×
...	...	...	

Precision:
fraction of correct responses among attempted responses ($\frac{2}{4}$)

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?	Right/Wrong?
`brunch`	yes	yes	✓
`dinner`	yes	–	×
`smog`	yes	yes	✓
`football`	yes	–	×
`brexit`	–	yes	×
...	...	...	

Precision:
fraction of correct responses among attempted responses ($\frac{2}{4}$)

Recall:
proportion of correct responses among true items (blends) ($\frac{2}{3}$)

Evaluation Metrics for Spelling Correction

We have some model predictions and a ground truth:

Candidate	Predicted blend?	True blend?	Right/Wrong?
brunch	yes	yes	✓
dinner	yes	–	×
smog	yes	yes	✓
football	yes	–	×
brexit	–	yes	×
...	...	...	

Precision:
fraction of correct responses among attempted responses ($\frac{2}{4}$)

Recall:
proportion of correct responses among true items (blends) ($\frac{2}{3}$)

F-Measure:
harmonic mean of precisino and recall $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

Approximate String Search and Matching
COMP90049 COMP30018 Knowledge Technologies
N-Gram Distance
Evaluation

Different types of errors:

Candidate	Prd't blend?	True blend?	R/W?	
`brunch`	yes	yes	✓	**true positive**
`dinner`	yes	–	×	**false positive**
`smog`	yes	yes	✓	**true positive**
`football`	yes	–	×	**false positive**
`brexit`	–	yes	×	**false negative**
...	...	...		

Evaluation Metrics

Different types of errors:

Candidate	Prd't blend?	True blend?	R/W?	
`brunch`	yes	yes	✓	**true positive**
`dinner`	yes	–	×	**false positive**
`smog`	yes	yes	✓	**true positive**
`football`	yes	–	×	**false positive**
`brexit`	–	yes	×	**false negative**
...	...	...		

Precision: $\frac{true\ positives}{true\ positives + false\ positives} = \frac{2}{4}$

Recall: $\frac{true\ positives}{true\ positives + false\ negatives} = \frac{2}{3}$

Evaluation Metrics

Different types of errors:

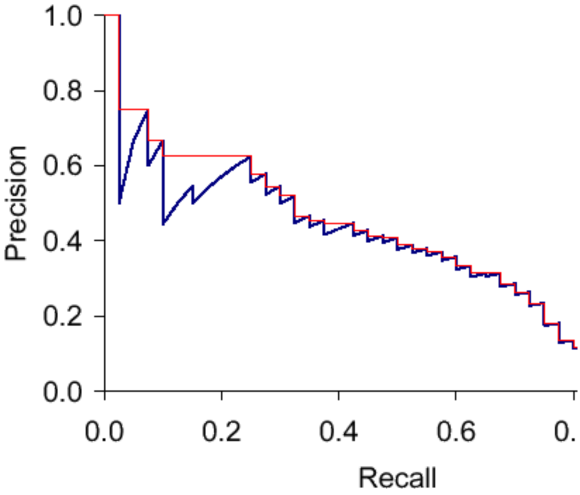
Candidate	Prd't blend?	True blend?	R/W?	
`brunch`	yes	yes	✓	**true positive**
`dinner`	yes	–	×	**false positive**
`smog`	yes	yes	✓	**true positive**
`football`	yes	–	×	**false positive**
`brexit`	–	yes	×	**false negative**
...	...	...		

Precision: $\frac{\textit{true positives}}{\textit{true positives} + \textit{false positives}} = \frac{2}{4}$

Recall: $\frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}} = \frac{2}{3}$

Evaluation Metrics

Different types of errors:



https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html

Comparing Systems

Typically, the value of the evaluation metric has little intrinsic meaning

Comparing Systems

Typically, the value of the evaluation metric has little intrinsic meaning

“This system gets 81% precision” — useful for users, or not?

Comparing Systems

The evaluation metric allows us to compare systems:

"The system based on the Global Edit Distance gets 81% precision, whereas the system based on the N-Gram Distance gets 84% precision"

Comparing Systems

The evaluation metric allows us to <u>compare</u> systems:

“The system based on the Global Edit Distance gets 81% precision, whereas the system based on the N-Gram Distance gets 84% precision”

“The basic system gets 81% precision, but after making some changes, the precision becomes 74%”

Comparing Systems

Typically, comparison is more difficult:

"System A gets 45% precision and 80% recall;
System B gets 95% precision and 10% recall"

Comparing Systems

Typically, comparison is more difficult:

"System A gets 45% precision and 80% recall;
System B gets 95% precision and 10% recall"

Which one should we use? Why?

Comparing Systems

Typically, comparison is more difficult:

“System A gets 45% precision and 80% recall;
System B gets 95% precision and 10% recall”

Which one should we use? Why?

The answer depends on the problem (and the user)!

Assignment 1!

Develop a Knowledge Technology that detects lexical blends in Twitter.

- Word blends are relatively novel words created by blending (not concatenating) two terms

breakfast	+	lunch	$\rightarrow$	brunch
fork	+	spoon	$\rightarrow$	spork
Britain	+	exit	$\rightarrow$	Brexit

- Social media is fertile grounds for linguistic innovation!

Your will develop knowledge about

- automatically detecting lexical blends among candidate tokens
- finding the original component words

Assignment 1!

Develop a Knowledge Technology that detects lexical blends in Twitter.

You will be given

- $\rightarrow$ demo folder!

You will be asked to

- find lexical blend candidates in the list of Twitter tokens (`candidates.txt`), using words from the English dictionary (`dict.txt`)
- evaluate your results against the reference set (`ref.txt`)
- critically analyze your methods and results
- optionally: use the full Tweet dataset (`tweets.txt`) for more advanced methods

Assignment 1!

Why is this a knowledge technology?

Assignment 1!

Why is this a knowledge technology?

- this is an open research problem – you have the chance to extend human knowledge
- we will never know the complete set of lexical blends (new ones occur all the time; very rare ones...)
- there is no perfect solution to detecting blends
- humans disagree on what a blend is (cf., evaluation set yourself!)
- tokens may be a word *and* a blend!
- most words are not blends (efficiency problem)
- different aspects of the task: detect blends? detect component words? Your choice!

Assignment 1!

Do s

- know your data
- do some literature research
- think scientifically, gain **knowledge**
- good writing (more next week)

Don't s

- try to achieve 100% precision
- (over-)optimize the software engineering part

Summary

- What is approximate string search?
- What are some common applications of approximate string search; why are they hard?
- What are some methods for finding an approximate match to a string? What do we need to generate them?
- How can we evaluate a typical approximate matching system?

Background Readings

Needleman, Saul B. and Wunsch, Christian D. (1970). “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. Journal of Molecular Biology 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4

(Originally in Russian, published in English as:) Levenshtein, Vladimir I. (1966). “Binary codes capable of correcting deletions, insertions, and reversals”. Soviet Physics Doklady 10 (8): 707–710.

Christin, P. (2006). “A Comparison of Personal Name Matching: Techniques and Practical Issues”. Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops

Background Readings

Kondrak, Grzegorz (2005). “N-Gram Similarity and Distance”. In Proceedings of the 12th international conference on String Processing and Information Retrieval (SPIRE’05), pp. 115-126, Buenos Aires, Argentina.

Peng, N. and Yu, M. and Drezde, M. (2015). “An Empirical Study of Chinese Name Matching and Applications”. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 377–383, Bejing, China.

Extension Readings

Whitelaw, Casey and Hutchison, Ben and Chung, Grace Y and Ellis, Gerard (2009). “Using the Web for Language Independent Spellchecking and Autocorrection”. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), pp. 890-899, Singapore, Singapore.

Ahmad, Farooq and Kondrak, Grzegorz (2005). “Learning a Spelling Error Model from Search Query Logs”. In Proceedings of the Human Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), pp. 955-962, Vancouver, Canada.