Detailed description of the Strudel algorithm

Marco Baroni

Abstract

This technical report provides a thorough description of the Strudel algorithm. Please see Baroni et al. (to appear) for a high-level discussion. We refer here to the specific version of Strudel used in the experiments of Baroni et al. (to appear) as Baked Strudel.

1 Introduction

Strudel works on an input corpus in which words have been automatically assigned part-of-speech tags and lemmas (with possible errors), and where the concepts to be analyzed have been marked by a special tag. Its output is a list of concept+property pairs, with a score indicating how important that property is for that concept, and the distribution of "generalized types" (see Section 5) connecting the concept and property in the corpus. For example, the input corpus might contain the following fragment:

```
CDV     NP      CDV
infection       NN      infection
of      IN      of
dogs    NNSCONCEPT      dog
was     VBD     be
ubiquitous      JJ      ubiquitous
30-60   CD      @card@
years   NNS     year
ago     RB      ago
```

The output list might contain the following properties for the concept of dog, where the third column is the association score and each couple of columns afterwards reports a type and the number of times the concept+target pair was connected by that type in the corpus (here, this type sketch is truncated to contain only types accounting for at least 10% of the overall type distribution of the concept+property pair):

```
dog kennel-n  1289.0852    in+left+n    89   for+right+n   25
dog owner-n    968.5315    of+right+n  173   's+left+n     59
dog bark-v     948.5887    _+left+v    694   _+right+v    357
```

2 Input

Strudel takes as input a stream of (lower-case) words with the corresponding part-of-speech tags and lemmas. We will refer to the inflected-form+tag+lemma tuples as tokens. Baked Strudel was trained on an input stream coming from the ukWaC corpus (that features automated part-of-speech tagging and lemmatization).[1] The target (nominal) concepts (i.e., the concepts Strudel will produce a representation for) must be marked with a special tag in the input stream. Appendix A reports the full Baked Strudel tagset with explanations and examples (useful to understand the examples below). Appendix B contains samples from the ukWaC input stream.

Optionally, lists of "keep" nouns, verbs and adjectives can also be provided as input (these words are selectively preserved in type patterns, as described in more detail in Section 3 below); our implementation also accepts keep adverbs, but, as they are not used in Baked Strudel, we ignore them here. The keep noun list of Baked Strudel was automatically extracted from the British National Corpus[2]. In particular, we extracted the top 50 trigrams of shape *function word + noun + function word*, and we picked the 48 (inflected) nouns that occurred in such trigrams as keep nouns (the rationale being that we wanted to identify nouns that tend to constitute parts of multi-word connectors in expressions such as *in a variety of*). Keep adjectives are simply the 10 most frequent adjectival lemmas in the BNC. Keep verbs are the 50 most frequent verbal lemmas.

Finally, Strudel accepts a stop list of adjectives, verbs and nouns that should not be considered potential properties. The stop list of Baked Strudel

[1]http://wacky.sslmit.unibo.it/
[2]http://www.natcorp.ox.ac.uk/

contains 10 very frequent nouns, 10 very frequent adjectives and 5 very frequent verbs.

The target concept list, the keep lists and the stop list of Baked Strudel are available as part of the supplementary online materials of Baroni et al. (to appear).[3]

3 Property extraction

While processing the corpus stream, whenever a target concept lemma is detected, the algorithm looks at windows of minimally 0 and maximally n tokens with the target as the leftmost or rightmost element, without crossing sentence boundaries. Baked Strudel uses windows of 7 tokens, since in informal experiments we found that, when using larger spans, we harvested virtually identical patterns. Given the "pattern grammar" we are about to describe, well-formed connectors of more than 5 tokens are extremely unlikely. If the window parameter is set to a suitably large value, it becomes irrelevant to the Strudel procedure.

Strudel analyzes each left/right window in which the token at the other end with respect to the target is a noun, verb or adjective (unless the corresponding lemma is in the stop list). We refer to these edge tokens as potential properties. The analysis is based on the tokens immediately preceding and following the window (henceforth called the left and right sides, respectively) and on what we call the connector pattern. The connector pattern is built by concatenating 4 parts: a token string, the concept position flag, the concept tag and the property tag. The token string is constructed by concatenating the tokens in the window minus the edges, i.e., minus the target and the potential property. However, before concatenation, tokens that contain nouns, verbs or adjectives are stripped off of the inflected form and lemma parts, unless they are in the corresponding keep lists (see Section 2, above). The token string can be empty (target and potential property are adjacent). The concept position flag is simply *left* or *right*, depending on the position of the target concept in the window (leftmost or rightmost token). The third and fourth segments of the connector pattern are the parts-of-speech of the concept and that of the potential property (always in this order). For example, given the stream:

[3] `http://www.cogsci.rpi.edu/CSJarchive/Supplemental/index.html`

```
bathroom NNCONCEPT bathroom
at IN at
the DT the
end NN end
of IN of
the DT the
corridor NN corridor
```

where *bathroom* is marked as a concept, *corridor* is the potential property under analysis, and *end* is in the keep noun list, the corresponding connector pattern is:

```
at/IN/at_the/DT/the_end/NN/end_of/IN/of_the/DT/the+left+NN+NN
```

This connector pattern encodes the following information: The sequence of tokens between the concept and the property is: *at* (with POS IN and lemma *at*), *the* (with POS DT and lemma *the*), *end* (with POS NN and lemma *end*), *of* (with POS IN and lemma *of*), *the* (with POS DT and lemma *the*). The concept is to the left of the property. The POS of the concept is NN (it is a singular noun). The POS of the property is also NN.

On the other hand, from the stream fragment:

```
provide VV provide
the DT the
latest JJS late
state NNCONCEPT state
```

and given that *late* is not in the keep adjective list, we extract the following connector pattern for the verb *provide* as a potential property of *state*:

```
the/DT/the_/JJS/+right+NN+VV
```

This connector pattern encodes the following information: The sequence of tokens between the concept and the property (in this case, with the property preceding the concept) is: *the* (with POS DT and lemma *the*), a superlative adjective (JJS) without lexical information. The concept is to the right of the property. The POS of the concept is NN, that of the property VV.

The connector pattern and the left/right sides are passed through a regular-expression-based pattern grammar that at the moment implements the following constraints (that should be interpreted in light of the tagset of Appendix A):

Global constraint

- The token string cannot contain commas, coordinating conjunctions, the existential *there*, foreign words, proper nouns, personal pronouns, adverbial particles, special symbols, interjections

Nominal property constraints

- Right side cannot be a noun, an adjective, the possessive particle or a pronoun
- If the token string is not empty, it must contain at least a preposition, or a pre-determiner, or the possessive particle, or a verb other than *be*, or a possessive pronoun
- The token string can contain maximally one noun
- If the token string contains a noun, the noun must be in the keep noun list
- If the token string contains a noun, there must be a preposition both before and after the noun

Adjectival property constraints

- The right side cannot be an adjective, the possessive particle or a pronoun
- The token string cannot contain punctuation marks, determiners, prepositions, nouns, pronouns, *to*, full verbs, wh-elements
- If the adjective precedes the target concept, the token string must be empty (adjective and noun must be adjacent)
- If adjective follows concept, token string must contain *be*

Verbal property constraints

- The token string cannot contain punctuation marks or full verbs

- If verb precedes concept, the right side cannot be a noun, an adjective, the possessive particle or a pronoun
- If verb precedes concept, the token string cannot contain *have*, *be*, or wh-elements
- If concept precedes verb and the verb is in the past tense or past participle form, the right side cannot be *to*
- If concept precedes verb, the token string cannot contain prepositions, nouns, adjectives, pre-determiners, the possessive particle, comparative or superlative adverbs

If the conditions on the left/right sides and connector pattern are satisfied, the concept+potential-property+connector-pattern tuple is added to the concept+property+pattern tuple list.

Sample input stream fragments and the corresponding output tuples are presented in appendix B. See also Baroni et al. (to appear) for commented examples. The script `Strudel-parser.pl`, available in the supplementary online materials, implements this phase of the procedure, and can be inspected for further details.

4 Property scoring

From the concept+property+pattern tuple list, we create a table that records, for each concept+property pair in the list, the number of *distinct* tuples in which it occurs (i.e., the number of distinct legitimate patterns that connect the concept and property). All counts used in this phase (O_{11}, C, P and N) are taken from this table.

Given a specific concept-property pair with observed co-occurrence frequency O_{11}, total count C for the concept paired with any property, total frequency P of occurrences of the property paired with any concept, and a total count N of all occurrences of all pairs, we construct observed and expected frequency contingency tables as follows:

O_{11}	$O_{12} = C - O_{11}$
$O_{21} = P - O_{11}$	$O_{22} = N - C - P + O_{11}$

$E_{11} = \frac{CP}{N}$	$E_{12} = \frac{C(N-P)}{N}$
$E_{21} = \frac{(N-C)P}{N}$	$E_{22} = \frac{(N-C)(N-P)}{N}$

From these tables, we calculate the *log-likelihood ratio score* of a concept-property pair as follows (the i and j indices range over the rows and columns of the contingency tables):

$$\text{llr} = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

According to this formula, when $O_{ij} = 0$ (the measured pair does not occur in the corpus), we would need to take the logarithm of 0, that is not defined. Following standard practice (Evert, 2005), in such cases, we assume that the corresponding term of the summation has 0 as its overall value.

The log-likelihood ratio approximates a χ^2 distribution with one degree of freedom (Evert, 2005). Based on this approximation, in Baked Strudel we keep only those pairs with a log-likelihood ratio above 19.51 (corresponding to a probability below 0.00001 under the assumption of independence) and where $O_{11} > E_{11}$ (i.e., association is significantly above, not below chance).

We chose the log-likelihood ratio as concept-property association measure since it is widely used in computational linguistics and it has been shown to be robust against data sparseness, providing reasonable significance-of-association estimates also in the presence of low counts in the contingency table cells (Dunning, 1993).

The scoring phase is implemented using the UCS association measure library.[4]

5 Type sketching

The last step of the Strudel procedure takes as input the list of concept+property pairs that were retained after property scoring and all their occurrences in the concept+property+pattern tuple list (the list constructed as discussed in Section 3), and produces the *type sketch* of a pair, i.e., it appends to each pair the distribution of its occurrences with "generalized types", obtained by simplifying the full patterns in the tuples. The detailed patterns in the concept+property+pattern list are mapped to generalized types by stripping

[4]`http://www.collocations.de`

off the concept and property parts-of-speech (keeping only a code for the basic category of the property: noun, verb or adjective), and by mapping the token string part of the pattern (see Section 3 above) to a simpler string. The mapping rules depend on the part-of-speech of the property in the tuple. If a string meets the conditions of multiple rules, only the first one (in the order presented here) applies. Patterns that do not match any rule are skipped (again, rules must be interpreted in terms of the tagset in Appendix A).

Mapping of patterns linking nominal properties

- If token string contains wh-possessive pronoun, map to WP$
- If token string contains possessive particle, map to *'s*
- If token string contains *of*, followed by keep noun, followed by preposition, map to preposition lemma
- If token string contains at least one preposition, map to first preposition lemma
- If token string contains *such* and *as*, map to *such_as*
- If token string contains verb lemma, map to verb lemma
- If token string contains verb, skip
- If token contains pre-determiner, map to pre-determiner lemma

Mapping of patterns linking adjectival properties

- If adjective precedes target noun, map to empty string
- If token string contains *be* followed by adverb, map to *is_ADV*
- Map to *is*

Mapping of patterns linking verbal properties

- If verb precedes target noun and there is a preposition, map to preposition lemma

- Map to empty string

The patterns extracted in Appendix B are mapped to generalized types in Appendix C. High-level examples are presented in the main paper. See also the script `map-patterns-to-types.pl`, available with the supplementary online materials.

After mapping the pattern in each tuple to a generalized type (possibly skipping some tuples if the pattern does not match a rule), we count the co-occurrences of each concept+property pair (among those that passed the property scoring phase) with each generalized type. We append the distribution of a concept+property pair across the generalized types to the concept+property as its type sketch.

This is the last step of the Strudel algorithm.

6 Output

The list of pairs that passed the property scoring phase, together with their log-likelihood ratio and their type sketch, constitutes the output of the Strudel algorithm.

The Baked Strudel list is available as part of the supplementary online materials (in this version, only the generalized types that account for at least 10% of the type sketch distribution of a pair are preserved: for a version with the full distribution, please contact us).

References

M. Baroni, B. Murphy, E. Barbu and M. Poesio. Strudel: A corpus-based semantic model based on properties and types. To appear in *Cognitive Science.*

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1): 61-74.

S. Evert. 2005. *The statistics of word cooccurrences.* Ph.D. dissertation, Stuttgart University, Stuttgart.

A The ukWaC part-of-speech tagset

The ukWaC corpus was tagged with the TreeTagger.[5] This appendix lists the corresponding tagset with explanations and examples. The constraints on connector patterns and left/right sides described in Section 3 and the mapping rules of Section 5 are based on the parts-of-speech listed here.

Tag	*Explanation*	*Examples*
$	currency symbol	$
“	opening quotes	“
”	closing quotes	”
(	opening parentheses	(
)	closing parentheses	)
,	comma	,
:	connecting punctuation	:, -, . . .
CC	coordinating conjunction	and
CD	cardinal number	four
DT	determiner	a, the, this, that
EX	existential there	there
FW	foreign word	ante, de
IN	preposition	on, of
JJ	adjective	near
JJR	comparative adjective	nearer
JJS	superlative adjective	nearest
LS	list item marker	A, iv
MD	modal auxiliary	might, will
NN	common noun, singular	action
NNCONCEPT	common noun in target concept list, singular	book
NNS	common noun, plural	actions
NNSCONCEPT	common noun in target concept list, plural	books
NP	proper noun, singular	Thailand, Thatcher
NPS	proper noun, plural	Americas, Atwells
PDT	pre-determiner	all, both, half
POS	possessive particle	’, ’s
PP	personal pronoun	I, he

[5] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

Tag	*Explanation*	*Examples*
PP$	possessive pronoun	my, his
RB	adverb	chronically, deep
RBR	comparative adverb	easier, sooner
RBS	superlative adverb	easiest, soonest
RP	adverbial particle	back, up
SENT	sentence-final punctuation	.
SYM	symbol or formula	US$500, R300
TO	infinitive marker	to
UH	interjection	aah, oh, yes, no
VB	be, base	be
VBD	be, past tense	went
VBG	be, -ing	being
VBN	be, past participle	been
VBP	be, plural	are
VBZ	be, -s	is
VH	have, base	have
VHD	have, past tense	had
VHG	have, -ing	having
VHN	have, past participle	had
VHP	have, plural	have
VHZ	verb, -s	believes
VV	verb, base	believe
VVD	verb, past tense	believed
VVG	verb, -ing	believing
VVN	verb, past participle	believed
VVP	verb, plural	believe
VVZ	verb, -s	believes
WDT	wh-determiner	what, which, whatever, whichever
WP	wh-pronoun	who, that
WP$	possessive wh-pronoun	whose
WRB	wh-adverb	how, when, where, why

B Input and output of property extraction phase: Examples

The following are sample fragments of the ukWaC input stream, followed by the concept+property+pattern tuples extracted from Baked Strudel given these fragments. Each fragment is centered on a target concept, with maximally 7 preceding and following tokens (spanning the maximum window plus left/right sides). Notice that ukWaC has been automatically tokenized, tagged and lemmatized, and thus there are errors in the input (e.g., *glow* in "make it glow" tagged as a noun).

Input

```
For IN for
more JJR more
information NN information
on IN on
the DT the
Alex NP Alex
Rider NP Rider
books NNSCONCEPT book
, , ,
please VV please
visit NN visit
www.alexrider.com NN www.alexrider.com
. SENT .
...
QI NP Qi
FESTIVAL NP Festival
CLUB NP Club
Why WRB why
not RB not
join VV join
the DT the
artists NNSCONCEPT artist
after IN after
the DT the
```

```
concerts NNS concert
? SENT ?
...
Primate NP Primate
’s POS ’s
leases NNS lease
to TO to
the DT the
Dawsons/Cremornes NP Dawsons/Cremornes
of IN of
lands NNSCONCEPT land
in IN in
the DT the
territory NN territory
of IN of
Clonfeacle NP Clonfeacle
, , ,
Co. NP Co.
...
genetic JJ genetic
makeup NN makeup
was VBD be
modified VVN modify
to TO to
include VV include
the DT the
genes NNSCONCEPT gene
from IN from
a DT a
jellyfish NNSCONCEPT jellyfish
that WDT that
make VVP make
it PP it
glow NNCONCEPT glow
in IN in
the DT the
dark NN dark
. SENT .
```

Output

```
artist join-v the/DT/the+right+NNS+VV
artist concert-n after/IN/after_the/DT/the+left+NNS+NNS
land territory-n in/IN/in_the/DT/the+left+NNS+NN
gene makeup-n was/VBD/be_/VVN/_to/TO/to_/VV/_the/DT/the+right+NNS+NN
gene include-v the/DT/the+right+NNS+VV
gene jellyfish-n from/IN/from_a/DT/a+left+NNS+NNS
jellyfish gene-n from/IN/from_a/DT/a+right+NNS+NNS
jellyfish make-v that/WDT/that+left+NNS+VVP
glow dark-n in/IN/in_the/DT/the+left+NN+NN
```

C Mapping to types: Examples

The sample patterns extracted from the input in Appendix B are mapped to the following generalized types:

```
the/DT/the+right+NNS+VV                                => _+right+v
after/IN/after_the/DT/the+left+NNS+NNS                 => after+left+n
in/IN/in_the/DT/the+left+NNS+NN                        => in+left+n
was/VBD/be_/VVN/_to/TO/to_/VV/_the/DT/the+right+NNS+NN => to+right+n
the/DT/the+right+NNS+VV                                => _+right+v
from/IN/from_a/DT/a+left+NNS+NNS                       => from+left+n
from/IN/from_a/DT/a+right+NNS+NNS                      => from+right+n
that/WDT/that+left+NNS+VVP                             => _+left+v
in/IN/in_the/DT/the+left+NN+NN                         => in+left+n
```