Using News Tweets to Generate Paraphrase Resource.

[IST 664: Natural Language Processing]

Syracuse University, Spring 2019

Instructor: Dr. Lu Xiao

By:

Vaibhav Kumar

Ayush Pramod Kumar

Dhaval Sonavaria

Rahul Rathod

Content

1.Introduction

We surveyed two existing paraphrase sources viz. DIRT and Berant. It is inferred that the existing resources are not being updated regularly. We, therefore present an approach which guarantees that the resource will be constantly updated. Since, we are querying news tweets on a daily basis to generate binary paraphrase pairs.

Keeping that in mind, much exertion has been dedicated to distinguishing predicate paraphrase, some of which brought about discharging assets of predicate entailment or paraphrase. Two principle approaches were proposed so far as that is concerned; the main influences the likeness in contention dispersion over an expansive corpus between two predicates (for example [a]0 purchase [a]1 / [a]0 gain [a]1) (Lin and Pantel, 2001; Berant et al., 2010). The second methodology abuses bilingual parallel corpora, separating as paraphrase sets of writings that were made an interpretation of indistinguishably to unknown dialects (Ganitkevitch et al., 2013).

While these methods have produced exhaustive resources which are broadly used by applications, their precision is restricted. In particular, the principal approach may remove antonyms, that additionally have comparative contention dissemination (for example [a]0 raise to [a]1 / [a]0 tumble to [a]1) while the second may conflate different faculties of the remote expression. We apply our methodology to create a resource of predicate paraphrases, exemplified in Table (A) below.

$[a]_0$ introduce $[a]_1$	$[a]_0$ welcome $[a]_1$
$[a]_0$ appoint $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ pass away at $[a]_1$
$[a]_0$ hit $[a]_1$	$[a]_0$ sink to $[a]_1$
$[a]_0$ be investigate $[a]_1$	$[a]_0$ be probe $[a]_1$
$[a]_0$ eliminate $[a]_1$	$[a]_0$ slash $[a]_1$
$[a]_0$ announce $[a]_1$	$[a]_0$ unveil $[a]_1$
$[a]_0$ quit after $[a]_1$	$[a]_0$ resign after $[a]_1$
$[a]_0$ announce as $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ threaten $[a]_1$	$[a]_0$ warn $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ live until $[a]_1$
$[a]_0$ double down on $[a]_1$	$[a]_0$ stand by $[a]_1$
$[a]_0$ kill $[a]_1$	$[a]_0$ shoot $[a]_1$
$[a]_0$ approve $[a]_1$	$[a]_0$ pass $[a]_1$
$[a]_0$ would be cut under $[a]_1$	$[a]_1$ slash $[a]_0$
seize $[a]_0$ at $[a]_1$	to grab $[a]_0$ at $[a]_1$

Fig: Sample from ranked paraphrases.

A third methodology was proposed to reap paraphrase from numerous notices of a similar occasion in news articles.1 This methodology accept that different repetitive reports settle on various lexical decisions to depict a similar occasion. Despite the fact that there has been some work following this methodology (for example Shinyama et al., 2002; Shinyama and Sekine, 2006; Roth and Frank, 2012; Zhang and Weld, 2013), it was less comprehensively examined and did not bring about making rework assets.
Fig:
In this report we present a novel unsupervised strategy for regularly developing extraction of lexically divergent predicate reword sets from news tweets. We apply our procedure to make an asset of predicate paraphrases.

Analysis of the resource obtained after ten long stretches of obtaining demonstrate that the arrangement of summarizes achieves the precision of 60-86% at various dimensions of help. Correlation with existing assets demonstrates that, even as our asset is as yet little in requests of size from existing assets, it supplements them with nonconsecutive predicates (for example take [a]0 from [a]1) and paraphrases which are exceedingly setting explicit. As of the finish of May 2017, it contains 456,221 predicate matches in 1,239,463 unique settings. Our asset is consistently developing and is required to contain around 2 million predicate paraphrase within a year. Until it achieves a sufficiently expansive size, we will discharge a day by day update, and at a later stage, we intend to discharge an intermittent update.

2. Existing Resources

There are a few existing paraphrase resources that have different number of paraphrases and are developed using different approaches. Here are the three paraphrase resources that we have but we took the DIRT and Berant resources into consideration relatively more than the PPDB.

- **DIRT**
- **PPDB**
- **Berant**

a. DIRT (Discovery of Inference Rules from Text)

Developed by Patrick Pantel and Dekang Lin at the University of Alberta, DIRT is a consolidated result of calculation and subsequent information gathering. The calculation involves paraphrase articulations from the source making use of reliance tree through distributional hypothesis. The articulated parse tree identifies duplex connection between instances and decides if the results are comparable or not.

DIRT Paraphrase Collection - RTE Users

Participants*	Campaign	Version	Specific usage description	Evaluations / Comments
BIU	RTE5		We used the canonical DIRT rulebase version of Szpektor and Dagan (RANLP 2007), and considered top 25 rules.	Ablation test performed. Positive impact of the resource: +1.33% accuracy on two-way task.
Boeing	RTE5		Verb paraphrases	Ablation test performed. Negative impact of the resource on two-way task: -1.17% accuracy. Null impact of the resource on three-way task.
UAIC	RTE5		Use of DIRT relations to map verbs in T with verbs in H	Ablation test performed. Positive impact of the resource: +0.17% accuracy on two-way, +0.33% on three-way task.
BIU	RTE4		We used the canonical DIRT rulebase version of Szpektor and Dagan (RANLP 2007), and considered top 25 rules.	+0.9% on RTE-4 ablation tests
Boeing	RTE4	Original DIRT db	Elaborate T sentence with DIRT-implied entailments	precision/recall in RTE4: boeing run1: 67%/6%; boeing run2: 54%/30%
UAIC	RTE4		Use of DIRT relations to map verbs in T with verbs in H	Ablation test performed: +0.7% precision on two-way task.
Uoeltg	RTE4			*Data taken from the RTE4 proceedings. Participants are recommended to add further information.*
UAIC	RTE3		Use of DIRT relations to map verbs in T with verbs in H	Ablation test performed: +0.37% precision on two-way task.
UIUC	RTE3		Paired verb/argument patterns	

Fig: DIRT Paraphrase Collection-RTE Users

b. PPDB (Predicate Paraphrase Database):

The Paraphrase Database (PPDB; Ganitke et al., 2013) is a broad semantic re-source, comprising of a rundown of expression sets with (heuristic) certainty gauges. In any case, it is as yet hazy how it can be best utilized, because of the heuristic idea of the confidences and its fundamentally deficient inclusion.

Fig: Homepage of Paraphrase.org

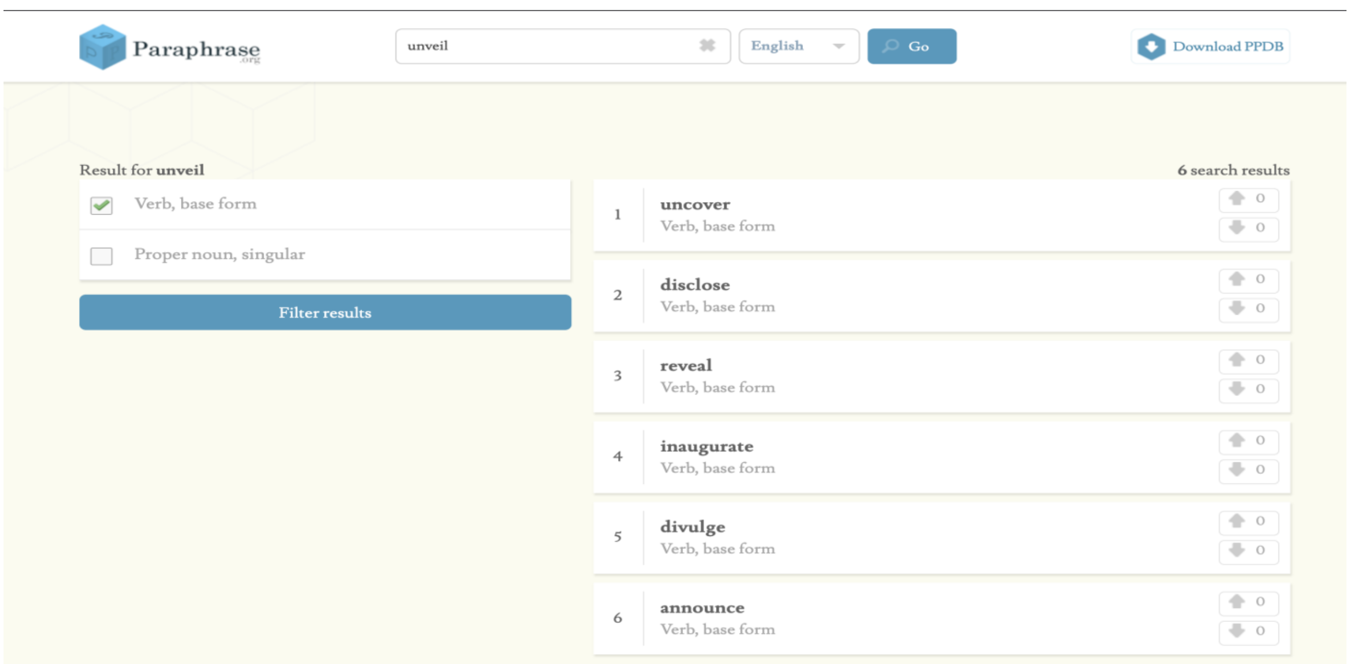

Fig: Paraphrase results from Paraphrase.org

c. Berant:

Berant 2012 built an entailment chart of distributionally comparable predicates by implementing transitivity imperatives and applying worldwide enhancement, discharging 52 million directional entailment rules (e.g. $[a]_0$ shoot $[a]_1$ -> $[a]_0$ murder $[a]_1$).

#	Template	Example	Question
1	$p.e$	`Directed.TopGun`	*Who directed Top Gun?*
2	$p_1.p_2.e$	`Employment.EmployerOf.SteveBalmer`	*Where does Steve Balmer work?*
3	$p.(p_1.e_1 \sqcap p_2.e_2)$	`Character.(Actor.BradPitt ⊓ Film.Troy)`	*Who did Brad Pitt play in Troy?*
4	$\texttt{Type}.t \sqcap z$	`Type.Composer ⊓ SpeakerOf.French`	*What composers spoke French?*
5	$\texttt{count}(z)$	`count(BoatDesigner.NatHerreshoff)`	*How many ships were designed by Nat Herreshoff?*

Table 1: Logical form templates, where p, p_1, p_2 are Freebase properties, e, e_1, e_2 are Freebase entities, t is a Freebase type, and z is a logical form.

	FREE917	WEBQUESTIONS
Our system	**73.9**	**41.2**
–VSM	71.0	40.5
–ASSOCIATION	52.7	35.3
–PARAPHRASE	31.8	21.3
SIMPLEGEN	73.4	40.4
Full matrix	52.7	35.3
Diagonal	50.4	30.6
Identity	50.7	30.4
JACCARD	69.7	31.3
EDIT	40.8	24.8
WDDC06	71.0	29.8

Table 6: Results for ablations and baselines on development set.

3.Data Source – News Tweets

We ae gathering news tweets from Twitter using Twitter API. The tweets are gathered daily and stored in folders acting as data source for the next step that is proposition extraction.

One of the many reasons of choosing news tweets, discussing same event, was the size limit of a tweet. Earlier it was 140-character limit for one tweet, which is now 280. Even though the size limit of each tweet has increased, it is still very safe to say that the data is concise. Increase in tweet size limit only caused the size of average tweet size to increase from 28-character per tweet to 33-character per tweet.

Another advantage of having daily news tweets as our data source is that the likelihood of the two tweets discussing same event on a given day is relatively high and it is justifiable to consider that the correctness of our implementation result will be much more high than done through any data source.

Following is an example collection of tweets collected on a same day. The level of similarity of the events can be inferred clearly.


CBSDenver @CBSDenver · 26/04/19
Deadly **Crash** and Fire on I-70: A video shows a white semi speeding past a runaway **truck** ramp moments before the pile-up and massive fire.
Investigators are working to confirm whether it's the same semi that caused the **crash**.
cbsloc.al/2Pzgq1W
2 4 4


Bloomberg @business · 07/04/19
Google and **Apple** were the first to **unveil** a "Netflix for video games," but Microsoft has the big-name games
The Big Problem for All the New Gaming Platforms
bloomberg.com
2 17 29


Jeff Gerstmann @jeffgers... · 25/03/19
Congrats to **Apple** for announcing a billion new shows for me to **claim** to have seen already just to avoid conversations with people who keep shouting YOU NEEEEEED TO SEE THIS SHOWWWWWWW at me.
24 298 1,976

4. Accuracy

Following are the Steps we have used after generating news tweets from Twitter.

1. Get verbal binary Dependency trees using Spacy.
2. Get predicates with binary arguments using PropS.
3. Rules to check paraphrases.
4. Create resource instance.
5. Append to Source.

Spacy:

We will extract dependency trees from the proposition we have acquired from the news tweet. Since, we are concerned with verbal, modal or auxiliary predicates, Spacy is a good choice because it generates spans of noun phrase, verbal phrase and prepositional phrase.

```
def np_chunk(self):
    """
    spaCy noun chunking, will appear as single token
    See: https://github.com/explosion/spaCy/issues/156
    """
    for np in self.toks.noun_chunks:
        np.merge(np.root.tag_, np.text, np.root.ent_type_)

    # Update mappings
    self.idx_to_word_index = self.get_idx_to_word_index()

def vp_chunk(self):
    """
    Verb phrase chunking - head is a verb and children are auxiliaries
    """
    self.chunk_by_filters(head_filter = lambda head: self.is_verb(head),
                          child_filter =lambda child: \
                          (self.is_aux(child) and len(self.get_children(child)) == 0) \
                          or (self.is_neg(child) and len(self.get_children(child)) == 0)\
                          or (self.is_prt(child) and len(self.get_children(child)) == 0))

def pp_chunk(self):
    """
    PP phrase chunking - head is a PP with a single PP child
    """
    def pp_head_filter(head):
        if not self.is_prep(head):
            return False
        children = self.get_children(head)
        if len(children) != 1:
            return False
        return self.is_prep(children[0])

    self.chunk_by_filters(head_filter = pp_head_filter, child_filter = lambda child: self.is_prep(child))
```

Verb phrases may exist as a single group of words or may be distributed across the sentence with a noun phrase inserted in between. Spacy provides heads and span matching with helps us generate a single verb phrase containing the proposition.

```
    def is_part_of_non_consecutive_span(self, head, tok):
        """
        Returns True iff tok is part of a non-consecutive span headed by head
        :param head: the head
        :param tok: a token
        :return: whether tok is part of a non-consecutive span headed by head
        """
        return (head in self.non_consecutive_spans) and (tok in self.non_consecutive_spans[head])

    def chunk_by_filters(self, head_filter, child_filter):
        """
        Meta chunking function, given head and children filters, collapses them together.
        Both head_filter and child_filter are functions taking a single argument - the node index.
        :param head_filter: head filter
        :param child_filter: child filter
        """
```

Generating predicates using PropS

After collecting tweets in a file and extracting verbal propositions we will now use rules to check is they are predicates.

We use two types of matching here:

Strict Matching

Load predicates and match arguments pairs to check if they are the same word. If this is true then the binary pair are a paraphrase instance.

WordNet argument matching:

If the .arguments are synonyms listed in the wordNet dictionary we also match them and create an instance.

```
# Find predicates that match by argument
predicate_alignments = pair_aligned_propositions(propositions, pronouns)

# Keep one tweet id pair and one (s,v,o) tuple for each instance
filtered = {tuple(sorted([tweet_id1, tweet_id2])):
                (tweet_id1, sent1, sf_pred1, pred1, sent1_a0, sent1_a1,
                 tweet_id2, sent2, sf_pred2, pred2, sent2_a0, sent2_a1)
            for (tweet_id1, sent1, sf_pred1, pred1, sent1_a0, sent1_a1,
                 tweet_id2, sent2, sf_pred2, pred2, sent2_a0, sent2_a1)
            in predicate_alignments}.values()

filtered = {(tuple([pred1, sent1_a0, sent1_a1]), tuple([pred2, sent2_a0, sent2_a1])):
                (tweet_id1, sent1, sf_pred1, pred1, sent1_a0, sent1_a1,
                 tweet_id2, sent2, sf_pred2, pred2, sent2_a0, sent2_a1)
            for (tweet_id1, sent1, sf_pred1, pred1, sent1_a0, sent1_a1,
                 tweet_id2, sent2, sf_pred2, pred2, sent2_a0, sent2_a1)
            in filtered}.values()

print 'Extracted %d instances' % len(filtered)
```

.

Argument Checking

After generating instances and confirming the arguments we create and filter candidates. The finals list of candidates to be declared as predicate paraphrases goes under another check using preposition to filter out the ones that may match arguments that are pronouns.

```python
def get_candidate_pairs(propositions, pronouns):

    global nlp

    # Remove propositions with argument pronouns
    propositions = [(tweet_id, sent, sf_pred, pred, a0, a1) for (tweet_id, sent, sf_pred, pred, a0, a1) in propositions
                    if len(pronouns.intersection(set([a0, a1]))) == 0 and len(sent) > 10]

    print 'Extracted %d propositions' % len(propositions)

    # Get candidates by lexical overlap in arguments
    candidates_by_args = defaultdict(set)

    [candidates_by_args[w].add(i) for i, (tweet_id, sent, sf_pred, pred, a0, a1) in enumerate(propositions)
     for w in a0.split() if not nlp.is_stop(w)]

    [candidates_by_args[w].add(i) for i, (tweet_id, sent, sf_pred, pred, a0, a1) in enumerate(propositions)
     for w in a1.split() if not nlp.is_stop(w)]

    # Get pairwise candidates from all lists
    candidates = set([propositions[i] + propositions[j] for lst in candidates_by_args.values()
                      for i in lst for j in lst if i != j])

    print 'Extracted %d candidates' % len(candidates)

    return candidates
```

To match the pronouns we use a pronouns list file and match them with the arguments.

Create Instances

Now after generating a candidate pair, we check instances daily and increase the count of the paraphrase if we see it daily. To add a heuristic element we also include the parameter of the number of days we began collecting tweets. Hence we use the following heuristic formula

$$s = count \cdot \left(1 + \frac{d}{N}\right)$$

Where: d : number of days in which the predicates were aligned.
N: Number of days since resource collection
count: Number of instances.

Screenshot of instances:

Candidate pairs:

8.39264E+17	{a0} met with {a1}	{a0} meet with {a1}	trump	russian ambassad	8.39264E+17	{a0} met {a1}	{a0} meet {a1}	trump	russian ambassador
8.39264E+17	{a0} make {a1}	{a0} make {a1}	people	quality content	8.39264E+17	were {a0} on {a1}	be {a0} on {a1}	people	his case
8.39264E+17	{a0} crashes into {a1}	{a0} crash into {a1}	train	charter bus	8.39229E+17	{a0} hits {a1}	{a0} hit {a1}	train	charter bus
8.39264E+17	{a0} might have hacke	{a0} may have hack into	the cia	phones	8.39264E+17	{a0} can hack {a1}	{a0} can hack {a1	the cia	your phone
8.39013E+17	leaving {a0} as {a1}	leave {a0} as {a1}	the country	row	8.38968E+17	leaving {a0} amid {	leave {a0} amid {	the country	an escalating row
8.39159E+17	in {a0} {a1}	in {a0} {a1}	our store	the 'ethical hackir	8.39102E+17	from {a0} {a1}	from {a0} {a1}	our store	the 'ethical hacking bundle
8.39264E+17	{a0} blames {a1}	{a0} blame {a1}	trump	obama	8.39264E+17	{a1} knew about {a	{a1} know about	alleged trump w	obama
8.39202E+17	{a0} released {a1}	{a0} release {a1}	wikileaks	thousands	8.39126E+17	{a0} has published {	{a0} have publish	wikileaks	thousands
8.39264E+17	{a0} crashes into {a1}	{a0} crash into {a1}	train	charter bus	8.39237E+17	{a0} hits {a1}	{a0} hit {a1}	train	bus
8.39159E+17	in {a0} {a1}	in {a0} {a1}	our store	the 'ethical hackir	8.39012E+17	from {a0} {a1}	from {a0} {a1}	our store	the ethical hacking bundle
8.39102E+17	from {a0} {a1}	from {a0} {a1}	our store	the 'ethical hackir	8.38936E+17	in {a0} {a1}	in {a0} {a1}	our store	the 'ethical hacking bundle
8.39264E+17	{a0} knew about {a1}	{a0} know about {a1}	obama	alleged trump wir	8.39264E+17	{a1} hits {a0}	{a1} hit {a0}	obama	trump
8.39073E+17	{a0} calls {a1}	{a0} call {a1}	ben carson	slaves	8.38921E+17	{a0} refers to {a1}	{a0} refer to {a1}	ben carson	slaves
8.39264E+17	{a0} accuses {a1}	{a0} accuse {a1}	trump	obama	8.39264E+17	{a0} hits {a1}	{a0} hit {a1}	trump	obama
8.39264E+17	{a0} lose {a1}	{a0} lose {a1}	only six millior	insurance	8.38926E+17	{a0} don't buy {a1}	{a0} do not buy {	people	insurance
8.39264E+17	{a0} get {a1}	{a0} get {a1}	people	health insurance	8.38926E+17	{a0} don't buy {a1}	{a0} do not buy {	people	insurance
8.39264E+17	{a0} unveil {a1}	{a0} unveil {a1}	republicans	bill	8.38947E+17	{a1} lets {a0}	{a1} let {a0}	republicans	this bill
8.39264E+17	{a0} has obtained {a1}	{a0} have obtain {a1}	wikileaks	trove	8.39264E+17	{a0} publishes {a1}	{a0} publish {a1}	wikileaks	massive trove
8.39264E+17	{a0} could lose {a1}	{a0} could lose {a1}	10 million peo	healthcare	8.39185E+17	{a0} have {a1}	{a0} have {a1}	people	health car
8.39264E+17	{a0} met {a1}	{a0} meet {a1}	donald trump	russian ambassad	8.39264E+17	{a0} met with {a1}	{a0} meet with {	trump	the russian ambassador
8.39264E+17	{a0} blames {a1}	{a0} blame {a1}	trump	obama	8.39264E+17	{a0} hits {a1}	{a0} hit {a1}	trump	obama
8.39073E+17	{a0} calls {a1}	{a0} call {a1}	ben carson	slaves	8.38906E+17	{a0} compared {a1}	{a0} compare {a	ben carson	slaves
8.39076E+17	{a0} benefit {a1}	{a0} benefit {a1}	local ihop don	easttnchildrens	8.39069E+17	{a0} go to {a1}	{a0} go to {a1}	donations	easttnchildrens
8.39264E+17	on {a0} {a1}	on {a0} {a1}	march	women	8.39074E+17	to highlight {a0} by	to highlight {a0}	work	women
8.39264E+17	{a0} released {a1}	{a0} release {a1}	wikileaks	thousands	8.39122E+17	{a0} publish {a1}	{a0} publish {a1}	wikileaks	1000s
8.39264E+17	{a0} make {a1}	{a0} make {a1}	people	mistakes	8.39264E+17	were {a0} on {a1}	be {a0} on {a1}	people	his case
8.39264E+17	{a0} criticizes {a1}	{a0} criticize {a1}	trump	obama 4 things	8.39264E+17	{a0} hits {a1}	{a0} hit {a1}	trump	obama
8.39012E+17	from {a0} {a1}	from {a0} {a1}	our store	the ethical hackin	8.38936E+17	in {a0} {a1}	in {a0} {a1}	our store	the 'ethical hacking bundle

After applying heuristic ranking:

	{a0} approve {a1}	{a0} pass {a1}	11569	544
	{a0} meet with {a1}	{a0} meet {a1}	11017	469
	{a0} say via {a1}	{a0} say {a1}	8233	715
	{a0} kill {a1}	{a0} shoot {a1}	8142	601
	{a0} ask {a1}	{a0} tell {a1}	7181	665
	{a0} tell {a1}	{a0} urge {a1}	6892	645
	{a0} get {a1}	{a0} have {a1}	6457	695
	{a0} have {a1}	{a0} take {a1}	6052	718
	{a0} tell {a1}	{a0} warn {a1}	6227	632
	{a0} get {a1}	{a0} receive {a1}	6143	650
	{a0} take {a1}	{a0} win {a1}	6383	501
	{a0} announce {a1}	{a0} unveil {a1}	5460	473
	{a0} get {a1}	{a0} sentence to {a1}	5635	415
	{a0} hit {a1}	{a0} strike {a1}	5516	397
	{a0} call {a1}	{a0} slam {a1}	4713	548
	{a0} blast {a1}	{a0} slam {a1}	4479	587
	{a0} accuse {a1}	{a0} slam {a1}	4558	550
	{a0} hit {a1}	{a0} reach {a1}	4488	526
	{a0} ask {a1}	{a0} urge {a1}	4238	538
	{a0} acquire {a1}	{a0} buy {a1}	4218	432
	{a0} die at {a1}	{a0} pass at {a1}	4893	263
	{a0} do not have {a1}	{a0} have {a1}	3527	622
	{a0} hit {a1}	{a0} rise to {a1}	3954	418
	{a0} climb to {a1}	{a0} rise to {a1}	4274	307
6.	{a0} call on {a1}	{a0} urge {a1}	3516	527

5. Resource Quality

We applied supervised learning on 100 pairs of paraphrases. These paraphrases have been taken from the highest accuracy bin.

Predicate Paraphrase Pair	Accuracy	Availability in PPDB
Need-Want	Accurate	Does not exist in PPDB
Close-Shut	Accurate	Exists in PPDB
kill-shoot	Accurate	Does not Exist in PPDB
Call-deny	Inaccurate	-
Celebrate-Mark	Inaccurate	-
beat-hold off	Accurate	Does not Exist in PPDB
send to-to deploy to	Accurate	Exists in PPDB
Flip-win	Inaccurate	-
not run for-not seek	Inaccurate	-
slam-tell	Accurate	Does not exist in PPDB
announce-launch	Accurate	Does not exist in PPDB
demand-want	Accurate	Does not exist in PPDB
pull out-withdraw from	Accurate	Exists in PPDB
accuse-call	Accurate	Does not exist in PPDB
accuse-blast	Accurate	Does not exist in PPDB
pull-remove	Accurate	Exists in PPDB
be on-go on	Accurate	Does not exist in PPDB

hike-raise	Accurate	Exists in PPDB
disclose-reveal	Accurate	Exists in PPDB
have-say	Inaccurate	-
give-offer	Accurate	Does not exist in PPDB
jail for-sentence to	Accurate	Exists in PPDB
become-will be	Accurate	Does not exist in PPDB
Have-suffer	Accurate	Does not exist in PPDB

Predicate Paraphrase Pair	Accuracy	Availability in PPDB
meet with - tell	Inaccurate	Does not exist in PPDB
be with - go to	Inaccurate	Does not exist in PPDB
remove - takedown	Accurate	Does not exist in PPDB
deliver - give	Accurate	Exists in PPDB
leak - reveal	Accurate	Does not exist in PPDB
claim - win	Inaccurate	Does not exist in PPDB
kill - murder	Accurate	Exists in PPDB
introduce - unwell	Inaccurate	Does not exist in PPDB
drop - plunge	Accurate	Does not exist in PPDB
jump to - rise to	Accurate	Does not exist in PPDB
hit - slam into	Accurate	Does not exist in PPDB
arrest - search for	Inaccurate	Does not exist in PPDB
have - need	Inaccurate	Exists in PPDB
separate from - take from	Accurate	Does not exist in PPDB
fall to - hit	Accurate	Does not exist in PPDB
call off - demand	Accurate	Does not exist in PPDB

call off - cancel	Accurate	Exists in PPDB

announce - confirm	Accurate	-
arrest - shoot	Inaccurate	Does not exist in PPDB
ask - call on	Accurate	Exists in PPDB
begin - start	Accurate	Exists in PPDB
blast - call	Inaccurate	Does not exist in PPDB
hold - keep	Accurate	Does not exist in PPDB
announce - declare	Accurate	-

Predicate Paraphrase Pair	Accuracy	Availability in PPDB
do - have	Inaccurate	Does not exist in PPDB
cancel - pull out	Accurate	Does not exist in PPDB
fire - launch	Accurate	Does not exist in PPDB
have - win	Inaccurate	Does not exist in PPDB
get - grant	Accurate	Does not exist in PPDB
say - slam	Inaccurate	Does not exist in PPDB
arrest - seek	Accurate	Does not exist in PPDB
accuse - lash out	Accurate	Does not exist in PPDB
put at - be at	Inaccurate	Exists in PPDB
pull - withdraw	Accurate	Exists in PPDB
have - want	Inaccurate	Does not exist in PPDB
win - go to	Inaccurate	Does not exist in PPDB
impose - slap	Inaccurate	Does not exist in PPDB

claim - take	Accurate	Does not exist in PPDB
climb to - hit	Inaccurate	-
award - win	Accurate	Does not exist in PPDB
blast - rip	Inaccurate	Does not exist in PPDB
accuse - sue	Accurate	Does not exist in PPDB
dismiss - reject	Accurate	Exists in PPDB
arrive in - visit	Accurate	Exists in PPDB
announce - say	Accurate	Exists in PPDB
lose - win	Inaccurate	Does not exist in PPDB
reveal - show	Accurate	Exists in PPDB
clinch - win	Inaccurate	Does not exist in PPDB
give - make	Inaccurate	Does not exist in PPDB

call- defend	inaccurate	-
get - receive	Accurate	Exists in PPDB
take-win	Inaccurate	-
announce - unveil	Accurate	Does Not Exists in PPDB
get - sentence to	Inaccurate	-
hit - strike	accurate	Does Not Exists in PPDB
call - slam	Inaccurate	-
blast - slam	Inaccurate	-
accuse - slam	Inaccurate	-
hit - reach	Inaccurate	-
ask - urge	Inaccurate	-
die at - pass at	Inaccurate	-

acquire - buy	accurate	Exists in PPDB
do not have - have	Inaccurate	-
hit - rise to	Inaccurate	-
call on - urge	accurate	Does not exist in PPDB
climb to - rise to	accurate	Exists in PPDB
die in - kill in	accurate	Does not exist in PPDB
quit as - resign as	accurate	Exists in PPDB
be sentence to - get	Inaccurate	-
have - make	Inaccurate	-
rip - slam	Inaccurate	-
reveal - share	accurate	Does not exist in PPDB
seek - want	accurate	Does not exist in PPDB
beat - defeat	accurate	Exists in PPDB
say- tell	accurate	Does not exist in PPDB

Final Statistics:

Total	Accurate	Inaccurate	in PPDB	not in PPDB
100	61	39	28	72

Therefore, the accuracy our data source achieves is 61% on the analysis of 100 instances.

6. Future Scope

We acquired fairly accurate predicate paraphrases from news tweets discussing the same event using the proposed new unsupervised method. We will release a growing resource of predicate paraphrases in the future, when the resource is comparable in size to the existing resources, since we generate a large number of paraphrases pairs, we sort them into four bins of increasing accuracy the smallest being the most accurate. Implement supervised learning, check paraphrase pairs before publishing paraphrase source.

References:

[1] Gabriel Stanovsky and Ido Dagan. Annotating and predicting non-restrictive noun phrase modifications. In ACL, 2016.

[2] Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. Getting more out of syntax with props. arXiv, 2016.

[3] Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. Automatic paraphrase acquisition from news articles. In HLT, pages 313–318. Morgan Kaufmann Publishers Inc., 2002.