A Supervised Approach To The Interpretation Of Imperative To-Do Lists

Paul Landes[1]
Barbara Di Eugenio[1]

[1]University of Illinois at Chicago
Department of Computer Science
Chicago, IL 60607, USA

Overview

Corpus

Corpus Pre-processing

Methods

Results

Definition

To-do lists are short utterances meant to remind the original author of a (usually concrete) task.

Example: *"Buy milk"*

Traditionally hand written and can include ordered steps. This work focuses on un-ordered tasks.

Introduction

- To-do lists are a popular medium for personal information management.
- Tracked in electronic form with mobile and desktop organizers–potential for software support for the corresponding tasks by means of intelligent agents [1, 5].
- Work in the area of personal assistants for to-do tasks, but no work focused on classifying user intention and information extraction.
- Our methods perform well across two corpora that span sub-domains.

Terms

- **Agent**: The user intention for the to-do task. *Think* The intelligent agent that resolves the task. Examples: *buy*, *find service*, *make appointment*.
- **Argument**: Tokens in the utterance that help the agent resolve the task. Examples: `date`=*Friday June 3rd* for the task *schedule meeting with Bob on Friday June 3rd*.

Example

- *"Buy swimsuit"*: `agent` = buy; `arguments`: item = *swimsuit*
- *"Call mom"*: `agent` = call; `arguments`: contact = *mom*
- *"Clean kitchen"*: `agent` = service; `arguments`: item = *kitchen*
- *"Book opera tickets"*: `agent` = find entertainment; `arguments`: item = *tickets*

Proprietary Corpus

- Corpus A was collected from an online publicly available source.
- The corpus was created by annotating each to-do task with an *intelligent agent* (IA), a respective set of *argument*s and an *item* argument category.
- Exceptions/errors were annotated for one of the following reasons:
  1. the to-do task itself is ambiguous
  2. language of the to-do task is not English
  3. illegal activity (i.e. *"buy drugs"*)
  4. professional or work-related
  5. meaningless language or gibberish

Proprietary Corpus Statistics

The Corpus A:

- Consisted of 3,169 annotations (one utterance per task).
- 1,342 were double annotated for inter-coder agreement (cohen's kappa).
- 3,169 divided into usable non-exceptions (1,690), and unusable exceptions (1,479).

Public Corpus

- Publicly available dataset composed of 102 volunteer contributed personal to-do tasks and 498 Trello to-do tasks with IA annotations.
- A subset of this data, including 68 volunteer and 218 Trello scraped to-do tasks.

Classification Process

1. Tokenize and chunk the utterance.
2. Part of speech (POS) tag.
3. Run Semantic Role Labeler (SRL).
4. Run Named Entity Recognizer (NER).
5. Classify agent A with trained model.
6. Use argument model A to extract tokens.

POS Tagger Model

Crucial *part of speech* (POS) tagger error: incorrectly tagging first token of utterances as non-verbs.

Use following criteria to reassign the POS tag of the first token:

1. identified as a present tense verb tag in WordNet[1] [7] and
2. identified as not a color, for example *"Green tea"* with "green" as a present tense verb.

[1] https://wordnet.princeton.edu

POS Tagger Model

First token model increased POS tagging accuracy up to 91.4% with an F-measure of 0.92 using the following features in addition to the aforementioned the heuristic method:

1. the POS tag of the first token
2. if it is a sentence containing one word
3. if there exists NER token spans greater than 1

Named Entity Recognition

- *Named entity recognition* (NER) provided additional context for classification.
- Two sets of features were created using both the NER [4] and the Stanford TokensRegex [2].
- Stanford TokensRegex [2] was enhanced to include a set of static word lists generated from Wikidata [8], Open Product Data [3], and the term lists
- The Wikidata lists were created with SPARQL queries (i.e. lists of foods, clothes, names of movies/video games).
- NER lists used to create token based regular expressions.

Agent Classification

Word count features

- Use the lemmatized form of the token for word count and cosine similarity features.
- Let $c_{wa} = \texttt{Count}(\texttt{w}, \texttt{a})$ be the count of word w for IA a and C_a be the set of word counts per IA such that $c_{wa} \in C_a$.
- Limit C to contain the highest n frequency counts with $n = |C_a|$ and hold n constant for all IAs as a hyper parameter.

Use the word count aggregation across C_a as feature:

$$WC_a = \sum_{c \in C_a} c \tag{1}$$

Significant performance gains were achieved by increasing n from 5 to 15 with the WC_a feature.

Agent Classification

Now we define a mapping from word to a word distribution over C normalizing by the word frequency:

$$q(w, a) = \frac{c_{wa}}{WC_a} \tag{2}$$

For example, for the $\texttt{buy}$ IA utterances *"Purchase a shirt. Iron shirt."*:

- $C_{buy} = \{c_{purchase} = 1, c_a = 1, c_{iron} = 1, c_{shirt} = 2\}$
- $q(purchase, \texttt{buy}) = 1/4$, $q(a, \texttt{buy}) = 1/4$, $q(iron, \texttt{buy}) = 1/4$, $q(shirt, \texttt{buy}) = 2/4$.

Agent Classification

- Word vector cosine distance was calculated with Word2vecJava [6].
- English Wikipedia pre-trained word vectors
- Sum over the token cosine similarity and weighting it with the word frequency distribution from equation 2.

Use MLE across all agents A to create cosine similarity (CS) feature for each sentence S:

$$CS_s = \underset{a \in A}{\operatorname{argmax}} \sum_{w_c \in C_a} \sum_{w \in S} q(w_c, a) \cdot \cos(w_c, w_s) \tag{3}$$

Argument Extraction

The model is trained first since it uses the argument classes as a features from the argument model.

Feature	Description
depend-label	SRL dependency of parent
head-depend-label	Proposition Bank argument
list-type	the term list attribute
ner-tag	Stanford's NER entity
next-pos	w_{n+1} POS tag
next-tm-ner-tag	w_{n+1} NER list entity
pos	w_n POS tag
prev-pos	w_{n-1} POS tag
stopword	if w_n a stop word
tm-ner-tag	our NER list entity
token-index	w_n sentence 0-index

Table 1: Argument features where w_n is the Nth word in the to-do task utterance.

Agent Classification Results

Id	Classifier	Features	Precision	Recall	F1
1	Baseline	N/A	0.10	0.31	0.15
2	LogitBoost	$verb + TNER$	0.57	0.55	0.51
3	NearestNeighbor	$CS_s + verb + TNER + NER + WC_a$	0.56	0.57	0.56
4	LogitBoost	$WC_a + verb + TNER$	0.67	0.66	0.65
5	LogitBoost	$CS_s + verb + TNER$	0.68	0.67	0.67
6	BayesNet	$CS_s + verb + TNER + NER + WC_a$	0.67	0.66	0.65
7	LogitBoost	$CS_s + verb + TNER + NER + WC_a$	0.70	0.70	**0.69**

Table 2: Agent classification results.

Argument Extraction Results

Agent	**Classifier**	**F1**
find-travel	AdaBoostM1	0.64
calendar	BayesNet	0.73
print	DecisionTable	0.79
find-activity	J48	0.78
self-improvement	J48	0.57
travel	JRip	0.90
call	KStar	0.88
plan-meal	KStar	0.68
find-service	NBTree	0.82
pay-bill-online	NBTree	0.85
text-sms	NBTree	0.83
search	NNge	**0.92**
contact	NaiveBayes	0.77
school-work	NaiveBayes	0.59
email	RandomForest	0.75
service	RandomForest	0.72
buy[1]	SMO	**0.72**

References

[1] Victoria Bellotti et al. “What a to-do: studies of task management towards the design of a personal task list manager”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2004, pp. 735–742.

[2] Angel X Chang and Christopher D Manning. *TokensRegex: Defining cascaded regular expressions over tokens*. Tech. rep. Technical Report CSTR 2014-02, Department of Computer Science, Stanford University, 2014.

[3] Open Product Data. *Open Product Data*. `http://product-open-data.com`. 2016.

[4] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. “Incorporating non-local information into information extraction systems by gibbs sampling”. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 363–370.

[5] Yolanda Gil et al. “Capturing Common Knowledge about Tasks: Intelligent Assistance for To-Do Lists”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 2.3 (2012), p. 15.

References (cont.)

[6] Andrew Ko. *Word2vecJava*. https://github.com/medallia/Word2VecJava. 2015.

[7] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[8] Wikidata. *Q5488768 — Wikidata*. http://www.wikidata.org/w/index.php?title=Q5488768&oldid=8079009. 2015.

Thank You!

Questions?