UTU at SemEval-2016 Task 10: Binary Classification for Expression Detection (BCED)

Jari Björne and Tapio Salakoski
Department of Information Technology, University of Turku
Turku Centre for Computer Science (TUCS)
Faculty of Mathematics and Natural Sciences, FI-20014, Turku, Finland
firstname.lastname@utu.fi

Abstract

The SemEval 2016 DiMSUM Shared Task concerns the detection of minimal semantic units from text and prediction of their coarse lexical categories known as supersenses. Our approach is to define this task as a binary classification problem approachable by straightforward machine learning methods.

We start by detecting semantic units by matching text spans against several large dictionaries, including the English WordNet, expressions derived from the Yelp Academic Dataset and concepts from the English Wikipedia, generating a set of potential supersenses for each matched span. For each potential supersense and text span pair a binary machine learning example is defined. We classify these examples using an ensemble method, taking as the final predicted supersense the one with the highest confidence score.

Our system achieves good performance on the supersense classification task but has limited performance for detection of multi-word semantic units. We show that the task of supersense prediction can be effectively defined as a binary classification task.

1 Introduction

The SemEval 2016 DiMSUM Shared Task[1] concerns the detection of *minimal semantic units* and their semantic classification using *supersenses* (Schneider et al., 2016). The minimal semantic units can consist of either single words or *multiword expressions (MWE)* in cases where multiple words form a lexical item. The concept of MWE as used in the DiMSUM task is described in detail in Schneider et al. (2014). The *supersenses* are broad-coverage, coarse lexical categories, 26 for nouns and 15 for verbs. These supersenses mostly correspond to the WordNet lexicographer files. A detailed description of the supersenses is in Schneider and Smith (2015). Automated supersense tagging has previously been explored by e.g. Curran (2005), Ciaramita and Altun (2006) and Johannsen et al. (2014).

In the DiMSUM task systems must both detect the minimal semantic units and assign correct supersenses for them. The final metric of the shared task is the averaged performance on these tasks, which are also evaluated independently. The systems can approach these tasks either independently (e.g. in a pipeline) or through a joint model.

We approach the DiMSUM task as a machine learning classification task where examples are generated for all semantic units found in a pre-defined dictionary. These semantic units are classified into one of the 41 supersenses by generating an example for each possible semantic unit + supersense pair, turning the task of assigning one of many supersenses into a binary classification problem. In this manner, the task becomes a generalized machine learning task for which almost any classifier can be used.

The DiMSUM shared task defines three subtasks or data conditions in which the systems can participate, varying in the amount of resources that are allowed to be used. The *supervised closed* condition defines the most limited case where participants can use the labeled training corpus, the English Word-

[1] http://dimsum16.github.io/

Proceedings of SemEval-2016, pages 850–855,
San Diego, California, June 16-17, 2016.

Net lexicon and two sets of Brown word clusters. In the *semi-supervised closed* condition the Yelp Academic Dataset can also be used, and in the *open* condition systems may use any and all resources available.

2 Data and Methods

2.1 Datasets

The DiMSUM corpus and evaluation tools, version 1.5, were provided by the task organizers. The corpus consists of a training and test set. The training set is a unified combination of the STREUSLE 2.1 corpus of web reviews, and the Ritter and Lowlands Twitter datasets, and consists of 4,799 sentences with tokenization, POS, MWE and supersense annotation. The test set consists of 1,000 sentences from online reviews, tweets and TED talk transcripts, with only tokenization and POS annotation.

The Yelp's Academic Dataset[2], which consists of JSON objects describing 13,490 businesses, 330,071 reviews and 130,873 users, was used with permission from Yelp Inc. WordNet version 3.0 (Fellbaum, 1998) was used through NLTK version 3.1 (Bird et al., 2009). For the English Wikipedia, the dump of titles in the main namespace *enwiki-20160113-all-titles-in-ns0.gz* was used[3]. Wikipedia page categories were downloaded using the Wikipedia Python library[4] by Jonathan Goldsmith.

2.2 System Overview

Our system follows a straightforward machine learning approach where examples are first generated and then classified using a standard classifier library. The task specific code is in the example generation part, which consists of two steps: 1) detection of minimal semantic units and 2) assignment of candidate supersenses for these units.

The detection of minimal semantic units is a rule-based step consisting mostly of dictionary matching. One sentence is processed at a time, one token at a time. For each token, the token itself and the following five tokens are first tested for a match, then the token and the next four tokens, and so on until just the token on its own. A set of *taggers* are applied for each span of tokens, and if any of the taggers finds a match, one or more examples are generated.

All tokens that are part of the matched set are then "consumed" and example generation continues from the next free token. In this manner, each token is assigned to the longest matched minimal semantic unit. As a consequence, each token may belong to a maximum of one candidate semantic unit. We do not attempt to detect disjoint MWEs.

A tagger will provide for each positive match a list of potential supersenses. A feature vector is built for each combination of the matched span of tokens and potential supersenses, with the example containing the correct supersense labeled as a positive and the rest as negatives.

2.3 Taggers

In our system taggers are the modules that detect minimal semantic units among the tokens in the sentence. For each set of consecutive tokens each tagger can generate zero or more candidate supersenses. Each unique candidate supersense will then be used to produce one example for classification.

The *WordNet tagger* is used in all three conditions as the primary tagger. Tested tokens are joined to a span by first using their lemmas and if no match is found by using their exact words. For the joined span all synsets are retrieved from WordNet and for each synset the corresponding lexicographer file name is added as a potential supersense. In the case of the *noun.Tops* file name the span itself is added as a supersense if it directly matches a known supersense.

The *Out-of-Vocabulary (OoV) tagger* is used in all three conditions to match semantic units not detected by the other taggers, providing a fuzzy matching system for common positive spans not included in any of the dictionaries. It detects such constructs as Twitter @-codes (*n.person*), possessive suffix tokens starting with an apostrophe (*v.stative*) and words whose DiMSUM supersense differs from their WordNet lexicographer file name, commonly businesses such as "restaurant", "store" or "hotel" (*n.group*). Cases for the OoV tagger were determined manually using the DiMSUM training corpus.

[2] https://www.yelp.com/academic_dataset

[3] https://dumps.wikimedia.org/enwiki/

[4] https://github.com/goldsmith/Wikipedia

2.3.1 The Yelp Tagger

The *Yelp tagger* is used in the *semi-supervised closed* and *open* conditions. First, three types of information are extracted from the Yelp Academic Dataset. For business objects the name of the business itself is used, but businesses also contain information on nearby schools and neighborhoods, and these are likewise added to the dictionary. When matching token sets, the businesses provide the *n.group* supersense and schools and neighborhoods the *n.location* supersense. Exact matches are tested for all of these cases.

Many business names have a common last noun, as in "Harvard Square Cafe" or "Minami Sushi". Thus, for token sets consisting only of nouns and proper nouns (POS tags *NOUN* or *PROPN*) a match is generated if the last token in the set matches the last token of any Yelp business or location. Candidate supersenses are generated based on all Yelp objects in which a token from the set is found in a corresponding position (first, middle or last). In addition to the *n.group* and *n.location* supersenses the additional supersense *n.food* is generated for the token "Restaurants".

Yelp user objects contain the first name and the last initial of the user. All first names are added to a dictionary. When detecting potential matches, first names of more than two characters are used as matches for the *n.person* supersense, tested against the first token in sets of no more than two uppercased tokens.

2.3.2 The Wikipedia Tagger

The *Wikipedia tagger* is used only in the *open* condition. All token sets for which a match cannot be provided by the WordNet tagger are compared against the list of Wikipedia page titles. Before matching the parenthesized disambiguation parts are removed from the titles. For matching titles, potential supersenses are provided according to the categories of that page. A page category can match zero or more DiMSUM supersenses.

Common Wikipedia page categories, present in pages for titles matching the DiMSUM training set, are manually assigned to supersenses if they are relevant for the DiMSUM task. For example, a category ending in "births" (e.g. "1968 births") corresponds to the *n.person* supersense, common media categories such as "album", "game", "television" or "comics" correspond to *n.communication* and any category ending in "companies" to *n.group*.

Based on these rules, the page categories of all matched Wikipedia titles are automatically mapped to supersenses. This mapping is done only for the subset of Wikipedia page titles found in the entire DiMSUM corpus, as processing the large Wikipedia dataset is quite time consuming. In this way, the resulting subset of 1,110 page titles and categories linked to corresponding DiMSUM supersenses can also be easily distributed alongside our source code.

2.4 Feature Representation

For each candidate token set + supersense pair a feature vector is built for machine learning. The same feature representation is used in all three DiMSUM task conditions.

For each token in the set a feature is built for the *lemma*, *POS* and *word* values. For the first and last tokens in the set, additional copies of these features are built, marked with the token's position. If the set consists of only a single token, an additional copy of these features is likewise built, marking that they come from a single-token example.

For the whole set of tokens, the consecutive *lemma*, *POS* and *word* values of the entire set are catenated together as features. The supersense of the token set + supersense pair is added as a feature. The supersenses of all the other examples generated for the same token set are also added as features distinct from the supersense feature of the current pair.

2.5 Machine Learning

For machine learning we use the scikit-learn library version 0.16.1 (Pedregosa et al., 2011). After examples are generated for each token set + supersense pair a binary classifier is used to classify them as either positives or negatives. We classify the examples with the Extra Trees Classifier ensemble method (Geurts et al., 2006) using its *predict_proba* function which gives probability estimates. For each token set, the final prediction is the supersense with the highest probability estimate among all supersenses predicted as positive for that token set.

3 Results and Discussion

The official results for the six teams participating in the DiMSUM shared task are shown in Table 1. Using the primary metric of the shared task, macro-averaged performance over both the minimal semantic unit detection and supersense assignment tasks, our system (BCED) was ranked third out of four participants in the *supervised closed* condition, fourth out of four participants in the *open* condition and was the only participating system in the *semi-supervised closed* condition.

3.1 Minimal Semantic Units

Compared to the other systems, the major limitation in our system was in the detection of the minimal semantic units. Our approach of detecting known MWEs, even using a very large dictionary such as the Wikipedia, was not sufficient to cover the scope of the DiMSUM MWEs, which consist of not only compound words and common idioms, but also more generalized noun phrases.

The numbers of generated examples are shown in Table 2. The vast majority of examples are detected by the WordNet tagger. It's notable how few additional positives are detected with the Wikipedia and Yelp dictionaries, even though the Wikipedia tagger considers only examples missed by the WordNet tagger. The number of examples detected by the OoV tagger doesn't change much between the conditions, as it mostly matches special cases not in any of the dictionaries.

The primary issue with the dictionary matching approach is that approximately a third of real positive examples are missed in all three conditions. The addition of first the Yelp and then the Wikipedia taggers decreases this number by 392 examples but compared to the total missed this is a relatively small improvement. Looking at the categories of the missed examples, MWEs that are *too long* (longer than six tokens) and those with *gaps* fall outside the scope of the system. Inclusion of the Yelp and Wikipedia taggers reduces the token sets for which *no match* is found by a considerable 718 examples, but there is a corresponding increase of 64 missing *type* examples for which a correct candidate supersense was not found.

Finally, the *nested* category highlights an interesting issue in example generation. Even while the Yelp and Wikipedia taggers provide more matches, at the same time the number of examples that are undetectable due to being nested within a false, longer MWE increases by 262. The issue of missing nested examples could be solved by generating examples for matched nested spans, but the corpus annotation might not be compatible with this approach, as each token can belong to a maximum of one annotated semantic unit. In the case of strong MWEs, for example in the idiom "close call", it is not clear what, if any, supersense the "call" token alone could be assigned (Schneider et al., 2014). However, in an MWE such as "cricket bat" (*n.artifact*) it is clear what the meaning of "bat" is, and likewise, "cricket" refers to the sport. Nevertheless, there is no annotation within MWEs, so while "cricket" alone has the supersense *n.act*, nested in "cricket bat" it has no supersense of its own. Thus, examples generated for nested spans would be inconsistent with non-nested examples.

	Super	Semi	Open
WordNet +	14435	14363	14294
WordNet −	78007	77045	76229
Wikipedia +	-	-	348
Wikipedia −	-	-	2635
Yelp +	-	334	329
Yelp −	-	675	640
OoV +	856	848	848
OoV −	756	751	751
total +	15203	15408	15595
total −	78714	78400	80158
too long	6	6	6
no match	3206	2876	2488
gaps	426	426	426
type	2783	2768	2847
nested	830	970	1092
total missed	7251	7046	6859

Table 2: Example generation for the training corpus. Each column corresponds to one of the conditions. The uppermost rows show the positive and negative examples generated by each of the four taggers, followed by the combined totals. After these are shown the five categories of false negative examples followed by their totals.

3.2 Supersense classification

While MWE detection performance was very low due to the issues in example generation, supersense

SYS	Team	Condition	μ-M	μ-S	μ-C	Macro-C	extra resources
214	ICL-HD	open	56.66	57.55	57.41	57.77	Yago3, GloVe embeddings
227	VectorWeavers	open	38.49	51.62	49.77	49.94	Google News EB, TurboParser
249	UW-CSE	open	57.24	57.64	57.57	57.71	Schneider MWE lexicons
255	BCED	open	13.48	51.93	46.64	47.13	English Wikipedia
211	BCED	semi-closed	13.46	51.11	45.86	46.17	
106	UFRGS	super-closed	51.48	49.98	50.22	50.27	
108	WHUNlp	super-closed	30.98	25.14	25.76	25.71	
248	UW-CSE	super-closed	53.93	57.47	56.88	57.10	
254	BCED	super-closed	8.20	51.29	45.47	45.79	
263	UW-CSE	open (late)	56.71	57.72	57.54	57.66	Schneider MWE lexicons

Table 1: The results of the DiMSUM 2016 shared task (provided by task organizers). Micro- and macro-averaged F-scores are shown for (M) MWE detection, (S) supersense assignment and (C) their combination. EB refers to embeddings.

assignment worked quite well (See Table 1). For supersenses, our approach reached F-scores of around 51% while the best performing systems were at 57%. Thus, our system placed 2nd and 3rd in terms of supersenses in the *supervised closed* and *open* conditions. As only 19% of all annotated minimal semantic units have more than one token the impact of MWE detection on supersense classification is limited.

The supersense classification is the only machine learning step in our system. Like most classifiers, the parameters of the Extra Trees Classifier must be optimized on known data for best performance. For parameter optimization, we used three-fold cross-validation on the training corpus, using one of the STREUSLE, Ritter and Lowlands datasets for performance estimation at a time and the other two for training.

Parameter optimization highlighted a conflict in using one binary classification step for both detecting MWEs and assigning the supersenses. Generally, the performance of the Extra Trees Classifier can be increased by increasing the ensemble size, and we noticed such increases while testing sizes from the default of 10 estimators up to 100. However, while the overall classification performance increased, the classifier became increasingly likely to not take any chances with MWE examples, assigning them all as negative. Only by decreasing ensemble size to 2 it became possible to detect at least some MWEs, albeit at the cost of slightly reduced performance on supersense assignment.

4 Conclusions

We developed a binary classification system for the DiMSUM 2016 task of detecting minimal semantic units and their supersenses. Our system consists of a customizable dictionary matching step for example generation, followed by a classification step.

The main advantage of this system is its simplicity. Standard machine learning systems can be applied for the classification of the generalized binary examples. As only one machine learning step is used, parameter optimization can likewise be performed in a standard cross-validation loop. The dictionary matching step can be quickly extended by adding new taggers for different vocabularies.

The primary shortcoming of the system is its low performance on MWE detection. This is largely a result of the dictionary matching approach being only partially applicable for the task of detecting the DiMSUM corpus MWEs. We speculate that using machine learning to follow the annotated scope of the MWEs would be a better approach for this part of the task.

Nevertheless, our system achieved good performance on supersense classification, indicating that binarizing this multi-class task is a valid approach. Moreover, even with a simple feature representation consisting only of token attributes and their combinations, comparatively high performance could be achieved.

We publish all of our experimental code, our Wikipedia derived datasets and our detailed results as an open source project [5].

[5] https://github.com/jbjorne/DiMSUM2016

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics.

James R. Curran. 2005. Supersense tagging of unknown nouns using semantic similarity. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 26–33, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.

Anders Johannsen, Dirk Hovy, Héctor Martınez Alonso, Barbara Plank, and Anders Søgaard. 2014. More or less supervised supersense tagging of twitter. *Proc. of* SEM*, pages 1–11.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nathan Schneider and Noah A Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T Mordowanec, Henrietta Conrad, and Noah A Smith. 2014. Comprehensive annotation of multiword expressions in a social web corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*

Nathan Schneider, Dirk Hovy, Anders Johannsen, and Marine Carpuat. 2016. SemEval 2016 Task 10: Detecting Minimal Semantic Units and their Meanings (DiMSUM). In *Proc. of SemEval*, San Diego, California, USA, June.