Team Taurus at SemEval-2019 Task 9: Expert-informed pattern recognition for suggestion mining

Nelleke Oostdijk
Centre for Language Studies
Radboud University
Nijmegen, The Netherlands
N.Oostdijk@let.ru.nl

Hans van Halteren
Centre for Language Studies
Radboud University
Nijmegen, The Netherlands
hvh@let.ru.nl

Abstract

This paper presents our submissions to SemEval-2019 Task9, Suggestion Mining. Our system is one in a series of systems in which we compare an approach using expert-defined rules with a comparable one using machine learning. We target tasks with a syntactic or semantic component that might be better described by a human understanding the task than by a machine learner only able to count features. For Semeval-2019 Task 9, the expert rules clearly outperformed our machine learning model when training and testing on equally balanced testsets.

1 Introduction

In the field of natural language processing, approaches featuring machine learning (ML) nowadays predominate. These have been shown to be quite effective with a wide range of tasks, including text mining, authorship attribution, and text classification. They are particularly suited for dealing with large data volumes and are robust in the sense that they can handle quite 'noisy' data. Unlike expert-informed approaches where rules need to be hand-crafted and apply only under predefined conditions, ML approaches learn from training data and are able to extrapolate. Proponents of ML approaches tend to dismiss the enterprise of hand-crafting rules as difficult, error-prone, time-consuming, and generally ineffective as even an extensive set of complex rules is bound to be incomplete and difficult to maintain.

Over the past years we have conducted a number of studies directed at the extraction of actionable information from microblogs. These include a range of topic areas and domains, including the detection of threatening tweets (Oostdijk and van Halteren, 2013a,b), the identification of potentially contaminated food supplements in forum posts (Oostdijk et al., submitted), and topic/event detection in tweets about natural disasters (floods (Hürriyetoğlu et al., 2016), earthquakes (Hürriyetoğlu and Oostdijk, 2017; Oostdijk and Hürriyetoğlu, 2017)), about traffic flow (Oostdijk et al., 2016) and about outbreaks of the flu (Hürriyetoğlu et al., 2017). In each case we have been exploring the role of the human expert in an expert-informed pattern recognition approach and a comparable ML approach, seeking out the strengths and weaknesses of either and attempting to arrive at a superior hybrid approach.

Semeval-2019, Task 9, Suggestion Mining (Negi et al., 2019), appeared to be another task that would lend itself to human rule building. As suggestions may be phrased in many different ways, successfully recognising that an utterance contains a suggestion requires human understanding of the context. Also, the amount of available training data was quite limited so that bringing into play the human expert's knowledge of the forms that suggestions may take would be an advantage, even though the task was not too clearly defined.

As a counterpart to the human rules, we built a machine learning system. For easier comparison of the patterns suggested by the machine learner and by the human expert, the learning component was a rather simple odds-based technique which still proved competitive in VarDial2018 (van Halteren and Oostdijk, 2018). As features we used character and token n-grams as well as syntactic patterns. In addition, the machine learner was somewhat expert-informed, as it was provided with several word lists related to suggestions.

Below we first describe the rule systems built by the human expert (Section 2) and the machine learner (Section 3) in some more detail. Then we proceed to the quantitative evaluation (Section 4), followed by a qualitative analysis of the evaluation phase (Section 5). We conclude with a discussion of what we learned in this shared task (Section 6).

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1247–1253
Minneapolis, Minnesota, USA, June 6–7, 2019.

2 Recognition with expert rules

The expert-rule-based system we applied uses a dedicated (task-specific) lexicon and set of hand-crafted rules in order to generate search queries. The system only targets suggestions, i.e. we only specify rules for patterns that should identify suggestions. Furthermore, we chose to design and apply one and the same set of rules and lexicon for both evaluation sets.

The lexicon comprises 1256 entries. Most of the entries are single words. However, we also included some multi-word items such *in order to*, *for example*, *at least* and *be able to* as well as phrase-like items such *why not* and *how about*. The lexicon includes a few frequently observed spelling variants, for example *pls*, *plz*, *dont* and *kinda*. With each entry, part-of-speech-like information is provided (e.g. verb, adjective, adverb but also *please*, which is given its own class). Typical examples of some of the lexicon entries are:

awesome	ADJ	would	AUXwould
just	ADV	very	ADVintens
provide	Vimp	provide	Vinfin

The rule set consists of 138 finite state rules. They describe how lexical items can combine to form search queries made up of multi-word n-grams. To give some examples, `AUXwould - BE - ADVintens - ADJ` matches e.g. *would be very helpful* and *would be really awesome*; `AUXshould - ADV - Vinfin` matches e.g. *should directly see* and *should properly support*; and `PLEASE - Vimp` e.g. *please provide* and *pls fix*. Rules typically combine two to five elements (parts of speech; POS). Given that some lexical entries are multi-word items, the patterns actually describe strings up to a length of nine words (one of the strings matching the rule `AUXcould - ADV - BE - ADVintens - ADJ` is, for example, *could at the very least be even more robust*).

In previous tasks, pattern recognition using word n-grams has been proven to be very effective. Moreover, the specification of the rules and the lexicon is much easier and far less time-consuming than would be the case for an all-encompassing description, as human experts need not concern themselves with describing elements and details irrelevant to the task at hand. In this particular case, trying to recognize suggestions, we experienced the drawback of only having access to the raw text input, something that did not bother us as much with other tasks. What we found was that in order to be able to recognize the many suggestions that take the form of an imperative sentence, we somehow needed to be able to distinguish between an imperative verb form and an infinitive or present tense verb. Noting that imperative verbs tend to occur (near) sentence- or clause-initially, we tried to account for this in our rules. In some cases we could make use of the presence of a comma, semi-colon or colon, that would identify the (potential) beginning of a clause. In such cases the punctuation mark was included as an element in the rules describing imperative structures. In order to identify the beginning of a sentence, we decided to automatically insert markers in the input during the preprocessing phase. These markers would also indicate the type of sentence (interrogative `MRKques` vs declarative or imperative `MRKsint`). Here the distinction between interrogatives and other types of sentence helped in distinguishing between *do/don't/dont* as an operator (auxiliary verb) in a question and as an operator in an imperative sentence. `MRKsint - Vimp` matches e.g *Allow applications to define ...* and *Ask for a room in ...*; and `MRKsint - ADV - Vimp` matches e.g. *At least support ...* and *Just don't forget ...*

3 Recognition by machine learning

For the ML approach we tried to use quite a wide spread of feature types, namely (a) character n-grams, with n ranging from 3 to 9; (b) token unigrams; (c) token n-grams, with n ranging from 2 to 4 (in which the individual positions can be filled with the actual token, the POS tag, or the word group (see below)); (d) syntactic structure of the main clause; (e) syntactic rewrites of all constituents; (f) syntactic n-grams (i.e. selected subtrees from the complete parse tree, e.g. function and category of a mother and two daughters).

For the character n-grams and the tokens, we applied a minimal level of pre-processing. In the training data, we cleaned up misplaced quotes and commas in the provided .csv-files. In the (later) trial data and in the evaluation data, which came in an HTML-format, we also replaced SGML-entities by their character counterpart, e.g. `&quot;` was replaced by `"`.

The POS tags and syntactic structure were produced by the Stanford NLP system (Manning et al., 2014). The dependency parse labeling was

then transformed to a constituency tree that conformed (as much as feasible) to our own view of English syntactic structure, being that developed in the TOSCA project (Aarts et al., 1998). All syntactic features were derived in either a fully lexicalized or fully unlexicalized version. Although we feel the syntactic features can be of considerable value, we consider this component a weak spot in the current system, as the parser regularly produces incorrect analyses. In the current task, this was especially the case for the training data.

In the lexicalized syntactic features, as well as in the token n-grams, a token slot could also be filled with a word group indicator. We found candidates for these lists by searching the 2006 Google n-gram collection (Brants and Franz, 2006) with some regular expressions, and then cleaned them up manually. There were four word groups. `GrpADJgood` contains 100 positive adjectives, e.g. *better* and *advisable*, being the first 100 manually approved from 1774 sorted matches with e.g. `^it (would|could|might|may) be (.*) (to|if)`. `GrpADVinten` covers 37 intensifying adverbs, e.g. *very* and *critically*, selected from 448 matches with e.g. `^(would|could|might|may) be (.*) ($adjgood) (to|if)`. `GrpVadvise` contains 723 verbs that indicate what is being suggested, e.g. *adopt* and *edit*, selected from 987 matches with e.g. `^(i|we) (suggest|advise|propose|request) you to (.*)`. And `GrpADVdescr` has 41 adverbs that can modify the advised action, e.g. *always* and *never*, selected from 212 matches with e.g. `^(suggest|advise|propose|request) that you (.*) ($vadvise)`. When used in a feature, each group indicator is concatenated with the POS tag. The introduction of the word groups appears to be effective: one of the strongest markers, found 82 times with suggestions and only twice with non-suggestions, is "It would be GrpADJgoodJJ".

Our choice of recognition system was influenced by the desire to compare to the human rules. After successful recognition, we wanted to be able to identify which features contributed to the success. For this experiment, we chose a very simple algorithm. We counted the occurrences of each feature in the training items marked as suggestions or as non-suggestions and compared the two counts to derive odds. For example, the character 8-gram `Please a` was found 60 times in suggestion items and once in non-suggestion items. Correcting for the different numbers of items in the two classes, and adding one to avoid division by zero, this led to odds of 45.89 in favour of suggestions. In order to avoid exaggerated counts because of repeated items, we only used the first occurrence of each item, leaving us with 1758 suggestions and 5339 non-suggestions. This removal was done even if the repeated items had different suggestion labels. For the actual recognition, we only used features that had odds higher than or equal to 3:1. In the prediction phase, the odds of all features present in an item were taken and their sum was compared to a threshold, which we chose by tuning on the trial data.

As with other tasks, we investigated hybridization of our two approaches. In this case, straightforward combination of the final choices seemed not very fruitful on the trial data. However, when we compared the suggestions for each individual feature type in the machine learning with those of the expert rules, we found especially syntactic n-grams were able to identify suggestions not found by the rules, which were limited to contiguous n-grams. We therefore built a combination that marked those items as suggestions that were recognized as such by the rules, plus those for which the recognition score by only the syntactic n-gram features was over the optimal threshold for the trial data. The relative quality of rules and machine learning on the trial data for Subtask B made us decide not to attempt combination there.

4 Quantitative evaluation

In this discussion, we do not want to compare to other systems in the shared task. For this we refer the reader to the task description paper (Negi et al., 2019), where it can be seen that more intricate machine learning systems, especially those using pre-trained language models, perform much better. We will rather examine how our internally comparable systems behave on the four item sets. In order to make the measurements compatible, we first have to make some adjustments. Both trial sets contained equal numbers of suggestions and non-suggestions, so that precision and recall were equally valuable. In the evaluation sets, there were more non-suggestions than suggestions, so that precision has more influence than recall. For comparison we needed to recalculate precision (and hence F_1; recall is unchanged) by

Approach	Meas	TrialA	EvalA	EvalBalA	TrialB	EvalB	EvalBalB
Rules	Prec	0.8213	0.4521	0.8761	0.9673	0.8669	0.8991
	Rec	0.8851	0.7586	0.7586	0.8045	0.7299	0.7299
	F_1	0.8520	0.5665	0.8132	**0.8784**	**0.7925**	**0.8057**
Learn	Prec	0.7914	0.4848	0.8898	0.5802	0.5099	0.5873
	Rec	0.8716	0.7356	0.7356	0.9134	0.8851	0.8851
	F_1	0.8296	**0.5848**	0.8054	0.7096	0.6471	0.7061
Combo	Prec	0.8179	0.4558	0.8778		NA	
	Rec	0.8953	0.7701	0.7701		NA	
	F_1	**0.8548**	0.5726	**0.8204**		NA	
Baseline	Prec	0.5872	0.1566	0.6141	0.7277	0.6877	0.7507
	Rec	0.9324	0.9195	0.9195	0.8267	0.7845	0.7845
	F_1	0.7206	0.2676	0.7364	0.7740	0.7329	0.7672

Table 1: Quality of submitted systems and organiser baseline.

Approach	TrialA	EvalA	EvalBalA	TrialB	EvalB	EvalBalB
Weighted sum	**0.8291**	0.6429	**0.8110**	0.7109	0.6637	0.7170
Char n-grams	0.7968	**0.6444**	0.7781	0.6782	0.6113	0.6748
Token 1-gram	0.7492	0.5411	0.7444	0.6006	0.5320	0.5563
Token 2-gram	0.8045	0.5882	0.7100	0.6006	0.5256	0.5436
Token 3-gram	0.7968	0.5342	0.6436	0.3955	0.3363	0.3420
Token 4-gram	0.7664	0.4348	0.5023	0.1201	0.1027	0.1029
Structure main clause	0.7580	0.4667	0.5588	0.3866	0.3029	0.3090
Syntactic rewrites	0.4888	0.3068	0.4454	0.1849	0.2085	0.2126
Syntactic n-grams	0.7601	0.5069	0.7297	**0.7419**	**0.6946**	**0.7416**

Table 2: F_1-score for each test set for each feature type, using model learned from training material for Subtask A, and oracle thresholds.

extrapolating the systems' behaviour to a balanced testset. E.g., the expert rules had 80 false accepts on a total of 146, so a precision of $66/146$ i.e. 0.4521. In a balanced dataset, 87 instead of 746 non-suggestions, not 80/746, but 9.3298/87 would be falsely accepted. Precision would be $66/(66 + 9.3298)$ i.e. 0.8761. We stress that we are not trying to make out results look more impressive, but merely want to make behaviour on trial and evaluation sets comparable.

Table 1 shows the evaluation results for our three systems and the organisers' baseline, with the adjusted scores shown in the columns marked EvalBal. Here, we can see that the enormous drops in quality between TrialA and EvalA were an illusion, caused by the difference in balance. Also, our three systems are now consistent in their order: expert rules outperform machine learning, but the (very eclectic) combination scores yet a bit better. In fact, we now see that all systems gain precision when going from TrialA to EvalA, but at the cost of recall. For the machine learner, this is not exclusively due to overtraining in threshold optimization, as we see below in the discussion of Table 2. A discussion of the results for the expert rules can be found in Section 5. The lower degree of change for the baseline can probably be explained by the rather general level of the rules there. Going from Subtask A to Subtask B, the machine learner suffers a substantial loss in quality, which is understandable as it is trained only on Subtask A data. Interestingly, there are almost no differences between TrialB and EvalB. For the expert rules, the situation is rather different. From TrialA to TrialB, we see a precision gain and recall loss, leading to a slight increase in F_1; but the move from TrialB to EvalB leads to a serious drop in both precision and recall.

In Table 2, we show the F_1-score for each individual feature type, as well as for the sum for comparison. On TrialA, the most useful individual feature types appear to be token bi- and trigrams and

Feature	TrainA	TrialA	EvalA	TrialB	EvalB
Total items	1758/ 5339	296/ 296	87/ 746	404/ 404	348/ 476
CG8_##Allow#	63/2	14/0	5/0	0/0	0/0
T4_WWWG_It_would_be_GrpADJgoodJJ	82/2	13/1	4/0	0/0	0/0
CG7_re#shou	31/1	8/0	0/1*	0/0	0/0
CG8_Please#a	60/1	6/0	2/0	0/0	0/0
SRFC_S_V_VP_A_VBx_OD_NP_A_PP	28/0	4/0	1/0	0/0	1/0
SCFFCCL_S_CS_AJP(nice)_A_SBAR(if)	20/1	4/0	1/0	1/0	0/1
CG8_#Provide	20/0	3/0	1/0	0/0	0/0
SCFFCCL_ROOT_UTT_S(suggest)_NOFUpunc_.(.)	14/0	3/0	1/0	4/0	5/0
T3_WWW_#_I_'d	24/0	4/1	0/3*	2/0	1/0
SCFFCCL_VP_AVB_MD(should)_MVB_VBN(GrpVadvise)	46/5	4/1	3/1	1/0	3/1*

Table 3: Ten high-odds features (excluding correlated ones), with effectiveness on all test sets. Each cell contains observation counts in suggestions/non-suggestions. Hash marks indicate spaces and out-of-sentence positions. The asterisk (*) means we disagree on one of the non-suggestions, but did use the provided labels.

character n-grams. The other feature types lag behind, but do help reach a higher score with the sum of all feature scores. Syntactic rewrites by themselves have a much lower F_1, but this is due to their low recall potential (precision 0.7267, recall 0.3682). When moving to EvalA, we see that character n-grams and token unigrams maintain their quality, indicating that the same kind of words are being used, but higher n-grams and syntax lose severely, which suggests that EvalA is more different from the training data than TrialA in how words are combined.

When we proceed to Subtask B, all feature types lose quality. As we moved to a different domain, and a different genre, this is not surprising. Machine learning depends on having training and test data that is as similar as possible. It is encouraging to see that the syntactic n-grams do manage to perform similarly to Subtask A. This means that machine learning at a more abstract level is able to move to another domain more easily.

If we examine which features are responsible for the recognition, we see that all play some role. There are, however, some more effective ones. We show ten of these in Table 3. Note that this is a manual selection, as simply taking the ten highest scoring ones would show only two basic patterns in various guises, e.g. *Please* as several character n-grams, as token unigram, in a token bigram, and as an adverbial in a syntactic n-gram. Some of the ten patterns need explanation: `CG7_re#shou` is part of the token bigram *there should*. `SRFC_S_V_VP_A_VBx_OD_NP_A_PP` means that a sentence is being rewritten as a verb phrase (i.e. a sequence of verb with possibly additional internal adverbials), followed by an adverbial realized by a non-finite verb, then a direct object realized by a noun phrase, and finally an adverbial realized by a prepositional phrase. If we search for examples, we find e.g. *Please allow the access to phone filesystem.*. It turns out that Stanford CoreNLP marks *Please* as a verb, placing *allow* as head of an "xcomp" clause, which confuses our analysis transformer and makes *allow* an adverbial. This is clearly wrong but, as it is done consistently, this pattern still provides a good marker. `SCFFCCL_VP_AVB_MD(should)_MVB_VBN(GrpVadvise)` indicates that within a verb phrase, we find both the modal *should* and a past participle of one of the verbs for something that is advisable. `SCFFCCL_S_CS_AJP(nice)_A_SBAR(if)` represents a sentence with a subject complement *nice* and an adverbial clause headed by *if*. And `SCFFCCL_ROOT_UTT_S(suggest)_NOFUpunc_.(.)` represents a sentence with a main verb with lemma *suggest*, combined with a period as punctuation, which nicely rules out questions with *suggest*. All these patterns have a good precision, but their recall is obviously limited. The first two manage to get about 5 percent on Subtask A, then we quickly drop to 2.5 and 1. In general, recall is similar for TrialA and EvalA. However, the strongest markers are absent altogether in Subtask B, where suggestions are apparently worded differently. Only the final two syntactic n-grams show a significant presence, which is in line with

the discussion on feature types above.

5 Qualitative evaluation

Upon inspection of the results obtained on the evaluation set with the expert rules, we specifically looked at the false accepts and false rejects. Contrary to what we thought might happen, only very few cases were missed out on due to omissions in the lexicon.

With cases that we missed (i.e. where we failed to recognize a suggestion) we did not find any clear clues as to what could be added to the patterns already specified in our rules set. There were some cases involving imperatives that we missed due to the fact that the punctuation mark we relied upon appeared to be absent, while gleaning the sentence type (interrogative) from the input final punctuation mark failed in cases where the input consisted of two or more sentences.

As for false accepts, we found that several cases were wrongly taken to be suggestions on the basis of a matching imperative pattern. Here the earlier problem of being unable to distinguish imperative verb forms from infinitives and present tenses no longer presented itself. Instead, word forms that are ambiguous between noun and verb such as *map* and *phone* were mistakenly held to be imperative verbs. Other false accepts were cases where otherwise highly successful rules proved to be too limited in their scope. For example, the pattern `AUXwould - Vinfin` would match *would allow* but then the sentence actually continues with *but* so that what initially looks like a suggestion turns out to be an observation, e.g. *The control would allow for the use of the contact picker but allows for manual entry and deletion of contacts*. Similar cases involving *but* were found with other patterns. In the cases of single instances slight modifications of the rules might have avoided wrongfully identifying something as a suggestion but there is very little evidence to go by, so there is no telling how it would impact on a different dataset.

We also noticed that some patterns occurred (almost) exclusively in Subtask A or B. Building separate rule sets for the two tasks might hence have been beneficial. However, the whole point of the exercise in Task 9 was to see whether a rule set can be ported to a new, mostly unknown, domain.

6 Discussion

The task was more difficult than we had expected. When we set out, we thought we knew what a suggestion was. However, after confronting our first version of the expert pattern system with the training data, we needed to review our ideas. After careful error analysis, we adapted our system and came to achieve quite acceptable results on the trial data, yet there remained a gap between what we (intuitively, as experienced language users) would consider a suggestion and quite a number of other cases which were labelled as such. In our view, there is a fine line between a suggestion and a non-suggestion, and perhaps one needs more context than a single sentence in order to tell which it is. We speculate that this may well explain the drop in performance. This is in line with the fact that the datasets provided, i.e. the training, trial and evaluation data, were rather different in nature: the training data comprised many inputs that were made up of multiple sentences, while the trial data and certainly the evaluation data mostly comprised single sentence inputs. Another factor which may have been at play here (and again we can only speculate), is that the annotations provided were made by different groups of annotators. This would then also explain some of the inconsistencies in the labeling, where similar cases were labeled differently.

Moving to our system, the expert-informed rule approach used previously with other tasks once more showed its strengths. The word n-gram patterns used are simple yet quite robust and effective, targeting specifically those parts of the input that are deemed relevant for the task at hand, without requiring these to be linguistically complete or well-formed phrases or clauses. There are no a priori limitations as to the length of the word n-grams. Compiling a lexicon and a set of rules requires limited effort. More than before, however, with the present task we experienced the limitations of using only contiguous word n-grams. Moreover, having only access to the raw text, information about the syntactic structure of a sentence was lacking, which in specific cases is key to being able to successfully identify a suggestion. We were able to fill some of this gap by combining with the ML approach, but improvements were as yet meagre as the parser was not performing optimally. Still, we have good hopes for this form of expert-informed hybridization.

References

Jan Aarts, Hans van Halteren, and Nelleke Oostdijk. 1998. The linguistic annotation of corpora: The TOSCA analysis system. *International journal of corpus linguistics*, 3(2):189–210.

Thorsten Brants and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

Hans van Halteren and Nelleke Oostdijk. 2018. Identification of differences between Dutch language varieties with the VarDial 2018 Dutch-Flemish subtitle data. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018), Santa Fe, New Mexico, USA*. Association for Computational Linguistics.

Ali Hürriyetoğlu, Nelleke Oostdijk, Mustafa Erkan Başar, and Antal van den Bosch. 2017. Supporting experts to handle tweet collections about significant events. In *International Conference on Applications of Natural Language to Information Systems*, pages 138–141. Springer.

Ali Hürriyetoğlu, Jurjen Wagemaker, Antal van den Bosch, and Nelleke Oostdijk. 2016. Analysing the role of key term inflections in knowledge discovery on twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the Web (KDWEB16). Cagliari, Italy*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Nelleke Oostdijk, Ali Hürriyetoğlu, Marco Puts, Piet Daas, and Antal van den Bosch. 2016. Information extraction from the social media: A linguistically motivated approach. In *Actes de la confrence conjointe JEP-TALN-RECITAL 2016, volume 10: Risque et TAL: 21-33. PARIS Inalco du 4 au 8 juillet 2016*.