NRC at i2b2: one challenge, three practical tasks, nine statistical systems, hundreds of clinical records, millions of useful features.

Berry de Bruijn, Colin Cherry, Svetlana Kiritchenko, Joel Martin, Xiaodan Zhu
Institute for Information Technology, National Research Council, Canada

Abstract

The team from National Research Council Canada, Institute for Information Technology, submitted systems for all three tasks within the 2010 I2B2 challenge. Each of the systems is built around a (semi-) supervised machine learning paradigm where elements of the source texts are represented as bags-of-features. The features were mostly derived from the text itself augmented with information from external sources: UMLS, cTAKES, and Medline. Our best scoring systems gave the following F-scores: task-1: 0.8523; task-2: 0.9362; task-3: 0.7313. We found that the greatest improvements for all three systems came from 'feature engineering' where external sources gave moderate improvements.

Introduction

The team from National Research Council Canada, Institute for Information Technology, submitted systems for each of the three tasks within the challenge. Given the differences between the three tasks, the approaches were quite different, although a number of characteristics are shared among the systems.

Each of the systems is built around a (semi) supervised machine learning paradigm where elements of the source texts are represented as bags-of-features. Patterns between features and desired ('ground truth') output in the training texts are learned by the system and this allows it to generate output for observed feature patterns in test texts. We can group the dimensions in the feature space as follows:

- surface features of that word (or token)
- concept mapping features of words and terms
- context features of that word
- features for the sentence in which the word occurs
- features for the section in which the word occurs
- features for the entire document

In most cases, the features are binary – i.e., either 'present' or 'absent' – and represented in a sparse vector – i.e., only 'present' features are written out.

Word surface features include: the word/token itself, whether it is a long/average/short word; whether it is all-uppercase, all-lowercase, or mixed-case; or all-digits or a mix of digits and letters; whether it contains punctuation, or is punctuation-only. They also include character-four-grams and word stems.

Concept mapping features are derived from existing annotation tools, namely cTAKES[1], MetaMap[2], and ConText[3]. Inspired by [4], we also use the Brown clustering algorithm[5] to create 7-bit hierarchical word clusters from the provided unlabeled data. Additionally, a few pattern matching algorithms from our own libraries were applied to cluster words and terms into more general concepts. These include: matching for negation words, auxiliary verbs, words indicating possibility or uncertainty, family members, past tense verbs, as well as terms from a symptom list, reaction word list, and preposition word list.

Context features include the token features of the neighbouring words with the window spanning up to 4 tokens before to 4 tokens after the word, as well as word bi/tri/quad-grams and skip-n-grams.

Sentence features include whether the sentence is long / average / short; upper/lower case letters were seen; digits were seen; sentence starts with an enumeration token; sentence ends with a colon; sentence contains possible / history / family / negation words; and whether verbs indicate past or future tense.

Section features: the (assumed) section headings are identified as the most recently seen all-caps line ending with a colon; the sub-section headings (if present) are assumed to be the most recently seen mixed-case line ending with a colon.

Document features include: upper-case / lower-case patterns seen across the document, and whether the document is long / average / short.

Task 1 system: design and performance

Task 1 concerns the identification of key concepts anywhere in the source text, and includes determining the exact boundaries of the concept, as well as the class of the concept ('problem', 'test', or 'treatment'). Concepts are non-overlapping and non-nested.

In our system, concept tagging is carried out using a discriminative semi-Markov HMM, trained using passive-aggressive online updates. Semi-Markov models[6] are Hidden Markov Models that tag multi-token spans of text, as opposed to single tokens. This allows us to conduct information extraction without devising a Begin/Inside/Outside (BIO) tagging formalism; instead, we need only four tags: *outside*, *problem*, *treatment*, and *test*. *Outside* is constrained to tag only single words, while the others can tag spans up to 30 tokens in length.

The semi-Markov model provides two major advantages over BIO. First, by labelling multi-token spans, labels cohere naturally. This allows the tagger to perform well without tracking the transitions between labels. Second, semi-Markov models allow much greater flexibility in feature construction, as one has access to the entire text of the concept as it is tagged, allowing easy inclusion of features such as concept length.

Semi-Markov models are generally trained as CRFs. However, we found CRF training to be too slow and memory intensive for our large feature sets. Instead, we train using an online algorithm similar to Collins' structured perceptron[7], called the Passive-Aggressive (PA) algorithm[8]. In particular, we use a loss-driven variant with a 0-1 cost. PA learning makes several passes through the training set. In each pass, each training example is visited once and decoded to find the max-loss response. The weight vector is then adjusted with the smallest possible update that will separate the correct tagging from the max-loss response by a margin of 1. During development, PA consistently outperformed a structured perceptron.

Table 1.1 shows the contributions of various feature groups, as measured by performance on a held-out development set drawn from all four hospitals, containing 3.1K sentences. We begin with a standard part-of-speech tagging feature set in (a), as described by [9]. This is augmented to indicate which features come from the first or last token in a concept, which is necessary for good boundary accuracy. In (b), we add word generalization features derived from lower-casing, word shape (the token "Chem-7" becomes "Aa-0"), and the 7-bit Brown clusters. We then add task-specific data in (c), in the form of section-heading features and features derived from the MetaMap and cTAKES taggers.

Up until this point, no features have required semi-Markov, multi-token functionality. We explore these

Feature Set	Rec	Prec	F
(a) Standard part-of-speech	.793	.832	.812
(b) + lowercase, shape, clusters	.807	.840	.823
(c) + headers, external taggers	.818	.848	.833
(d) + semi-Markov features	.826	.858	.842
(e) + tag trigrams w/ annotation	.829	.859	.844

Table 1.1: Concept tagging feature contributions.

features next in (d). Unfortunately, straightforward semi-Markov features, such as concept length, were not helpful. However, we found the following to be useful:

Bracket matching: indicates if a concept has mismatching round brackets

Concept sequence: using simplified MetaMap labels (designed to match the task concepts) and cTAKES chunk labels, we generate features indicating the sequence of labels completely contained within a proposed concept, so that "trace edema at ankles" becomes "*umls_problem umls_body_part*". This allows the system to learn, for example, that a body-part alone is unlikely to be a *problem*, but a concept containing a problem indicator followed by a body part is probably a *problem*.

Preposition counting: for concepts with three or more words, we include a feature that counts the number of prepositions in the concept. This is designed to capture the annotation standard, which encourages at most one preposition.

Function word sequence: similar to the concept sequence feature, we also include a function word sequence, which allows us to generalize a concept to patterns such as "the * with the *".

Finally, in (e) we add transition features in the form of tag bigrams and trigrams, such as "*test outside problem*". These were initially harmful; however, if we augment outside tags for common words with their lexical items, we can create meaningful tag *n*-grams, such as "*test out_for problem*", which are helpful.

Since the PA algorithm has no explicit regularization, it is helpful to average the parameters values over all updates[7]. We report results using this averaged weight vector. We used cross-validation to determine that 15 passes through the training data was sufficient for good performance. The complete system, which includes all of the features described above, trains in about 1.5 hours on modern hardware, and assigns 1.1M features non-zero weights. The effective feature space explored contains 4.2M features. We did not find over-fitting to be an issue.

For the official test runs, we selected three system variations: (1) the complete system; (2) a system that did not use UMLS or cTAKES features; (3) a system that included self-training on unlabelled data[10]. Table 1.2 summarizes performance on the test set.

The results demonstrate that the external sources of semantic and syntactic tagging (UMLS and cTAKES) are beneficial; together they improve F-measure by 1.5 percentage points. Unfortunately, bootstrapping on the unlabelled data (System 3) showed no improvement.

	True Pos	False Neg	False Pos	Recall	Prec	F-score
Exact						
Sys-1	37646	7363	5683	.8364	.8688	.8523
Sys-2	36776	8233	6125	.8170	.8572	.8366
Sys-3	37663	7346	5787	.8367	.8668	.8515
Inexact						
Sys-1	41339	3670	1990	.9167	.9322	.9244
Sys-2	40783	4226	2118	.9037	.9246	.9140
Sys-3	41421	3588	2029	.9184	.9305	.9244

Table 1.2: Test set performance for the three systems – for exact and inexact spans.

Task 2 system: design and performance

This task is phrased as follows: for every 'problem' concept in a text, assert whether that concept was found to be present, absent, possible, conditional, hypothetical, or associated with someone else.

We solve this task in two stages. In stage 1, we generate assertion class predictions for every word that is part of a 'problem' concept. In stage 2, a secondary classifier predicts a class for the complete concept based on the (various) per-word predictions.

In stage 1, each word is represented as a large, sparse, binary feature vector, as described in the Introduction. Three classifiers are trained and applied independently: (1) the SVM-multiclass from the SVM-light/SVM-struct library[11], which outputs one score per class per word; (2) LibSVM[12], where six classifiers are trained and applied in a one-vs-rest set-up, resulting in one score per class per word; and (3) an in-house multiclass passive-aggressive learner (see Task-1), outputting one score per class per concept. The stage 2 classifier used SVM-multiclass with a linear kernel, default parameter settings and a C-parameter value of 20,000, as selected using a development set.

Feature Set	F
(a) Words, word n-grams, character n-grams	.899
(b) + token/ sentence/ section/ doc features	.924
(c) + taggers: MetaMap / cTAKES / ConText	.937
(d) + all features from neighbouring words	.944

Table 2.1: Assertion task feature contributions.

The features used in stage 1 were virtually all features mentioned in the Introduction. All groups of features benefited performance: table 2.1 summarizes how different groups of features contributed to the F-score when sequentially inserted, as tested on one hold-out set.

Given this general architecture, we produced three variants for our official submission:

• System 1: the complete two-stage process, as described above.

• System 2: a simplified system. Stage 1 consisted of SVM-multiclass alone predicting word-level classes. Stage 2 remained unchanged.

• System 3: designed to improve minority class recall, even if it comes at the expense of reduced overall performance. The output of System 1 was overruled when the LibSVM score on 'associated-with-someone-else'-vs-rest exceeded a hand-set threshold for any of the words in a concept. The same was then done for 'hypothetical', 'conditional', and 'possible' – mimicking the order specified in the challenge guidelines. If none of the scores overruled System-1 output, the original output was retained.

	True Pos	False Neg	False Pos	Recall	Prec	F-score
Sys-1	17366	1184	1184	.9362	.9362	.9362
Sys-2	17338	1212	1212	.9347	.9347	.9347
Sys-3	17197	1353	1353	.9271	.9271	.9271

Table 2.2: Test set performance for the three systems.

System 1 achieved 93.62% accuracy. Table 2.3 shows the class-by-class confusion matrix between prediction and truth. The matrix shows that despite efforts to balance type-1 errors and type-2 errors, the classifier still tends to favour the majority class. Fishing out the 'conditional' cases was troublesome, with a recall only slightly above 15%. Also, the mislabelling of true 'possible' cases as 'present' accounted for 33% of our system's mistakes.

System 2, while being much simpler in design, gave by and large comparable results: an accuracy of 0.9347 and a similar contingency table (not shown). Most errors were caused by a yet stronger preference to label cases as 'present', the majority class. The higher System-1 score indicates that there is still some independency between its classifiers.

System 3 performed as expected: it increased recall (reflected by higher numbers on the diagonal of Table 2.4) for all the minority classes – including, in fact, 'absent' – even as precision dropped enough to lower micro-averaged accuracy to 0.9271. Average F-score, when calculated per class and then macro-averaged, is .774 for System 3, up from .753 for System 1 and .740 for System 2.

truth / *pred*	absent	AWSE	conditi-onal	hypo-thetical	possible	present
absent	**3370**	20	6	13	14	121
AWSE	3	**105**		1		1
cond			**26**			1
hypoth.	4			**617**	10	48
possib	14		1	15	**468**	74
present	218	20	138	71	391	**12780**

Table 2.3: System 1 performance confusion matrix; concept counts for *class predictions* (rows) and truths (columns). AWSE=associated-with-someone-else

truth / *pred*	absent	AWSE	conditi-onal	hypo-thetical	possible	present
absent	**3409**	9	5	12	21	273
AWSE	4	**124**		1		2
cond	1		**44**	2		30
hypoth.	4			**621**	11	53
possib	20			12	**491**	159
present	171	12	122	69	360	**12508**

Table 2.4: System 3 performance confusion matrix

Task 3 system: design and performance

The goal of task 3 is to determine the relationship between a pair of concepts provided that the two concepts appear in the same sentence and one (or both) of them is a 'problem' concept. Task 3 defines five categories of treatment-problem relations, two categories of test-problem relations, and one category of problem-problem relation.

We trained three separate classifiers to categorize treatment-problem, test-problem, and problem-problem relations respectively. The classification framework was maximum entropy (ME), and we used an OpenNLP ME toolkit[13]. Relations were classified independently; i.e., a decision made on one concept pair does not affect other decisions.

Our baseline feature set is similar to that of [14], which was further augmented with features derived directly from the concepts and assertion-tagged text and from the external MetaMap and cTAKES taggers, as described in the Introduction. For the convenience of later discussion, we call all these features augmented-baseline features. Note that the specific feature sets used by the three classifiers were different from each other, which were decided during development.

In addition, we found the following features and design decisions to be beneficial.

a) Exploiting parsing trees. We parsed the input texts using the Charniak parser with its improved, self-trained biomedical parsing model[15]. These were then transferred into Stanford dependencies[16]. Features extracted included words, their tags (e.g., POS tags), and arc labels on the dependency path between the minimal trees that cover the two concepts, along with the word and tags of their common ancestor, and the minimal, average and maximal tree distances to the common ancestor. We observed an additional 0.4 point gain in F-score on 5-fold cross validation on the training set.

b) Balancing category distribution. In the training set, some of the relationship types were observed much more often than others – e.g., there were about 8-times more negative problem-problem relations than positive ones. We addressed this issue by down-sampling the training set to a pos/neg ratio between 1:2 and 1:4, as selected using a development set. This reduced a classifier's bias towards the majority class, and improved the overall F-score by about 0.3 point on 5-fold cross validation. This was especially important when a system included unsupervised bootstrapping (as our System 3 did), since bias is amplified when bootstrapping is applied.

c) Using semantics features. We used Medline as a semi-structured source of knowledge, calculating Pointwise Mutual Information (PMI) between two concepts as found in Medline abstracts to approximate the relatedness of these concepts. This weak approximation to structured knowledge of concept relationships still yielded about 0.2 point improvement during development.

d) Semi-supervised training. We also applied bootstrapping on the provided unlabeled data. For this, our system for Task-1 was applied to the unlabeled documents to provide concept span tags and labels. This bootstrapping gave us about 0.4 point improvement. As we attempted self-training for all three tasks, it is interesting that it was successful only for Task 3. We suspect that this may be because Task 3 had the smallest amount of training data.

We submitted three system variants for Task 3. System 1 used the augmented-baseline features discussed above as well as the dependency features. System 2 additionally employed the PMI statistics from Medline collocations, and System 3 included bootstrapping on the provided unlabeled documents.

Table 3.1 shows the performance of each system variant on the final test set. The system improved its exact F-score with each version, with Medline PMI providing a 0.3 gain, and self-training providing an additional 0.4 gain.

	True Pos	False Neg	False Pos	Recall	Prec-	F-score
Exact						
Sys-1	6296	2809	1965	.6902	.7611	.7239
Sys-2	6269	2801	1896	.6911	.7677	.7274
Sys-3	6288	2782	1838	.6932	.7738	.7313
Clustered						
Sys-1	6704	2366	1522	.7391	.8149	.7752
Sys-2	6678	2392	1487	.7362	.8178	.7749
Sys-3	6641	2429	1485	.7321	.8172	.7723

Table 3.1: Test set performance for the three systems for exact class matches and clustered-class matches.

Conclusions

The team from NRC-IIT posted sound results for each of the three tasks in the 2010 i2b2 challenge. Our best scoring systems, built from supervised and semi-supervised statistical machine learning components, gave an F-score of 0.8523 on task 1, 0.9362 on task 2, and 0.7313 on task 3. Our choices in machine learning algorithms allowed us to expand the feature space without risking overfitting. A wide range of textual features gave more significant improvements while the gains from the knowledge-rich resources we applied -- MetaMap, cTAKES, and MedLine – seemed moderate.

Acknowledgments

Lynn Wei and Chengbi Dai provided help programming system components.

The 2010 i2b2/VA challenge and the workshop are funded in part by the grant number U54-LM008748 on Informatics for Integrating Biology to the Bedside from National Library of Medicine, and facilities of the VA Salt Lake City Health Care System with funding support from the Consortium for Healthcare Informatics Research (CHIR), VA HSR HIR 08-374 and the VA Informatics and Computing Infrastructure (VINCI), VA HSR HIR 08-204. MedQuist co-sponsored the 2010 i2b2/VA challenge meeting at AMIA.

References

1. Savova GK, Kipper-Schuler K, Buntrock JD, Chute CG. UIMA-based clinical information extraction system. LREC: Towards enhanced interoperability for large HLT systems 2008.
2. Aronson AR, Lang F-M. An overview of MetaMap: historical perspective and recent advances. J Am Med Inform Assoc. 2010;17:229-36.
3. Harkema H, Dowling JN, Thornblade T, Chapman WW. ConText: An algorithm for determining negation, experiencer, and temporal status from clinical reports. J Biomed Inform. 2009;Oct; 42(5): 839-51.
4. Miller S, Guinness J, Zamanian A. Name Tagging with Word Clusters and Discriminative Training. In HLT-NAACL 2004, Boston, USA.
5. Brown PF, Della Pietra VJ, deSouza PV, Lai JC, Mercer RL. Class-Based n-gram Models of Natural Language. Computational Linguistics, 1992;18(4):467–479.
6. Sarawagi S, Cohen WW. Semimarkov conditional random fields for information extraction. In ICML 2004.
7. Collins M. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In EMNLP 2002.
8. Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y. Online Passive-Aggressive Algorithms. JMLR 2006;7(Mar):551—585
9. Ratnaparkhi A. A maximum entropy part-of-speech tagger. In EMNLP 1996.
10. McClosky D, Charniak E, Johnson M. Effective Self-Training for Parsing. In HLT-NAACL 2006, Brooklyn, USA.
11. Joachims T. Making Large-Scale SVM Learning Practical. In: B. Schölkopf and C. Burges and A. Smola (ed.). Advances in Kernel Methods - Support Vector Learning. MIT Press, 1999.
12. Chang C-C, Lin C-J. LIBSVM : a library for support vector machines, 2001. Software available at www.csie.ntu.edu.tw/~cjlin/libsvm
13. http://maxent.sourceforge.net/index.html
14. Patrick J, Li M. A Cascade Approach to Extracting Medication Events. In: Proc. Australasian Language Technology Workshop (ALTA) 2009.
15. McClosky D. Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing. Ph.D. thesis, Brown University 2010.
16. de Marneffe M-C, MacCartney B, Manning CD.. Generating Typed Dependency Parses from Phrase Structure Parses. In LREC 2006.