A Clustering Approach for the Nearly Unsupervised Recognition of Nonliteral Language*

Julia Birke and Anoop Sarkar
School of Computing Science, Simon Fraser University
Burnaby, BC, V5A 1S6, Canada
jbirke@alumni.sfu.ca, anoop@cs.sfu.ca

Abstract

In this paper we present TroFi (Trope Finder), a system for automatically classifying literal and nonliteral usages of verbs through nearly unsupervised word-sense disambiguation and clustering techniques. TroFi uses sentential context instead of selectional constraint violations or paths in semantic hierarchies. It also uses literal and nonliteral seed sets acquired and cleaned without human supervision in order to bootstrap learning. We adapt a word-sense disambiguation algorithm to our task and augment it with multiple seed set learners, a voting schema, and additional features like SuperTags and extrasentential context. Detailed experiments on hand-annotated data show that our enhanced algorithm outperforms the baseline by 24.4%. Using the TroFi algorithm, we also build the TroFi Example Base, an extensible resource of annotated literal/nonliteral examples which is freely available to the NLP research community.

1 Introduction

In this paper, we propose TroFi (Trope Finder), a nearly unsupervised clustering method for separating literal and nonliteral usages of verbs. For example, given the target verb "pour", we would expect TroFi to cluster the sentence "Custom demands that cognac be *poured* from a freshly opened bottle" as literal, and the sentence "Salsa and rap music *pour* out of the windows" as nonliteral, which, indeed, it does. We call our method nearly unsupervised. See Section 3.1 for why we use this terminology.

We reduce the problem of nonliteral language recognition to one of word-sense disambiguation by redefining *literal* and *nonliteral* as two different senses of the same word, and we adapt an existing similarity-based word-sense disambiguation method to the task of separating usages of verbs into literal and nonliteral clusters. This paper focuses on the algorithmic enhancements necessary to facilitate this transformation from word-sense disambiguation to nonliteral language recognition. The output of TroFi is an expandable example base of literal/nonliteral clusters which is freely available to the research community.

Many systems that use NLP methods – such as dialogue systems, paraphrasing and summarization, language generation, information extraction, machine translation, etc. – would benefit from being able to recognize nonliteral language. Consider an example based on a similar example from an automated medical claims processing system. We must determine that the sentence "she hit the ceiling" is meant literally before it can be marked up as an ACCIDENT claim. Note that the typical use of "hit the ceiling" stored in a list of idioms cannot help us. Only using the context, "She broke her thumb while she was cheering for the Patriots and, in her excitement, she hit the ceiling," can we decide.

We further motivate the usefulness of the ability to recognize literal vs. nonliteral usages using an example from the Recognizing Textual Entailment (RTE-1) challenge of 2005. (This is just an example; we do not compute entailments.) In the challenge data, Pair 1959 was: Kerry hit Bush hard on his conduct on the war in Iraq. → Kerry shot Bush. The objective was to report FALSE since the second statement in this case is not entailed from the first one. In order to do this, it is crucial to know that "hit" is being used nonliterally in the first sentence. Ideally, we would like to look at TroFi as a first step towards an unsupervised, scalable, widely applicable approach to nonliteral language processing that works on real-world data from any domain in any language.

* This research was partially supported by NSERC, Canada (RGPIN: 264905). We would like to thank Bill Dolan, Fred Popowich, Dan Fass, Katja Markert, Yudong Liu, and the anonymous reviewers for their comments.

2 Previous Work

The foundations of TroFi lie in a rich collection of metaphor and metonymy processing systems: everything from hand-coded rule-based systems to statistical systems trained on large corpora. Rule-based systems – some using a type of interlingua (Russell, 1976); others using complicated networks and hierarchies often referred to as *metaphor maps* (e.g. (Fass, 1997; Martin, 1990; Martin, 1992) – must be largely hand-coded and generally work well on an enumerable set of metaphors or in limited domains. Dictionary-based systems use existing machine-readable dictionaries and path lengths between words as one of their primary sources for metaphor processing information (e.g. (Dolan, 1995)). Corpus-based systems primarily extract or learn the necessary metaphor-processing information from large corpora, thus avoiding the need for manual annotation or metaphor-map construction. Examples of such systems can be found in (Murata et. al., 2000; Nissim & Markert, 2003; Mason, 2004). The work on supervised metonymy resolution by Nissim & Markert and the work on conceptual metaphors by Mason come closest to what we are trying to do with TroFi.

Nissim & Markert (2003) approach metonymy resolution with machine learning methods, "which [exploit] the similarity between examples of conventional metonymy" ((Nissim & Markert, 2003), p. 56). They see metonymy resolution as a classification problem between the literal use of a word and a number of pre-defined metonymy types. They use similarities between *possibly metonymic words* (PMWs) and known metonymies as well as context similarities to classify the PMWs. The main difference between the Nissim & Markert algorithm and the TroFi algorithm – besides the fact that Nissim & Markert deal with specific types of metonymy and not a generalized category of *nonliteral* language – is that Nissim & Markert use a supervised machine learning algorithm, as opposed to the primarily unsupervised algorithm used by TroFi.

Mason (2004) presents CorMet, "a corpus-based system for discovering metaphorical mappings between concepts" ((Mason, 2004), p. 23). His system finds the selectional restrictions of given verbs in particular domains by statistical means. It then finds metaphorical mappings between domains based on these selectional preferences. By finding semantic differences between the selectional preferences, it can "articulate the higher-order structure of conceptual metaphors" ((Mason, 2004), p. 24), finding mappings like LIQUID→MONEY. Like CorMet, TroFi uses contextual evidence taken from a large corpus and also uses WordNet as a primary knowledge source, but unlike CorMet, TroFi does not use selectional preferences.

Metaphor processing has even been approached with connectionist systems storing world-knowledge as probabilistic dependencies (Narayanan, 1999).

3 TroFi

TroFi is not a *metaphor* processing system. It does not claim to interpret *metonymy* and it will not tell you what a given *idiom* means. Rather, TroFi attempts to separate literal usages of verbs from nonliteral ones.

For the purposes of this paper we will take the simplified view that *literal* is anything that falls within accepted selectional restrictions ("he was forced to eat his spinach" vs. "he was forced to eat his words") or our knowledge of the world ("the sponge absorbed the water" vs. "the company absorbed the loss"). *Nonliteral* is then anything that is "not literal", including most tropes, such as metaphors, idioms, as well phrasal verbs and other anomalous expressions that cannot really be seen as *literal*. In terms of metonymy, TroFi may cluster a verb used in a metonymic expression such as "I read Keats" as nonliteral, but we make no strong claims about this.

3.1 The Data

The TroFi algorithm requires a *target set* (called *original set* in (Karov & Edelman, 1998)) – the set of sentences containing the verbs to be classified into literal or nonliteral – and the seed sets: the *literal feedback set* and the *nonliteral feedback set*. These sets contain feature lists consisting of the stemmed nouns and verbs in a sentence, with target or seed words and frequent words removed. The frequent word list (374 words) consists of the 332 most frequent words in the British National Corpus plus contractions, single letters, and numbers from 0-10. The target set is built using the '88-'89 Wall Street Journal Corpus (WSJ) tagged using the (Ratnaparkhi, 1996) tagger and the (Bangalore & Joshi, 1999) SuperTagger; the feedback sets are built using WSJ sentences con-

Algorithm 1 *KE-train*: (Karov & Edelman, 1998) algorithm adapted to literal/nonliteral classification

Require: $\mathcal{S}$: the set of sentences containing the *target word*
Require: $\mathcal{L}$: the set of literal seed sentences
Require: $\mathcal{N}$: the set of nonliteral seed sentences
Require: $\mathcal{W}$: the set of words/features, $w \in s$ means w is in sentence s, $s \ni w$ means s contains w
Require: ϵ: threshold that determines the stopping condition

1: $w\text{-}sim_0(w_x, w_y) := 1$ *if* $w_x = w_y, 0$ *otherwise*
2: $s\text{-}sim^I_0(s_x, s_y) := 1$, for all $s_x, s_y \in \mathcal{S} \times \mathcal{S}$ where $s_x = s_y$, 0 otherwise
3: $i := 0$
4: **while** (**true**) **do**
5: $\quad s\text{-}sim^L_{i+1}(s_x, s_y) := \sum_{w_x \in s_x} p(w_x, s_x) \max_{w_y \in s_y} w\text{-}sim_i(w_x, w_y)$, for all $s_x, s_y \in \mathcal{S} \times \mathcal{L}$
6: $\quad s\text{-}sim^N_{i+1}(s_x, s_y) := \sum_{w_x \in s_x} p(w_x, s_x) \max_{w_y \in s_y} w\text{-}sim_i(w_x, w_y)$, for all $s_x, s_y \in \mathcal{S} \times \mathcal{N}$
7: $\quad$ **for** $w_x, w_y \in \mathcal{W} \times \mathcal{W}$ **do**
8: $\quad\quad w\text{-}sim_{i+1}(w_x, w_y) := \begin{cases} i = 0 & \sum_{s_x \ni w_x} p(w_x, s_x) \max_{s_y \ni w_y} s\text{-}sim^I_i(s_x, s_y) \\ \text{else} & \sum_{s_x \ni w_x} p(w_x, s_x) \max_{s_y \ni w_y} \{s\text{-}sim^L_i(s_x, s_y), s\text{-}sim^N_i(s_x, s_y)\} \end{cases}$
9: $\quad$ **end for**
10: $\quad$ **if** $\forall w_x, \max_{w_y}\{w\text{-}sim_{i+1}(w_x, w_y) - w\text{-}sim_i(w_x, w_y)\} \leq \epsilon$ **then**
11: $\quad\quad$ **break** # algorithm converges in $\frac{1}{\epsilon}$ steps.
12: $\quad$ **end if**
13: $\quad i := i + 1$
14: **end while**

taining seed words extracted from WordNet and the *databases of known metaphors, idioms, and expressions* (DoKMIE), namely Wayne Magnuson English Idioms Sayings & Slang and George Lakoff's Conceptual Metaphor List, as well as example sentences from these sources. (See Section 4 for the sizes of the target and feedback sets.) One may ask why we need TroFi if we have databases like the DoKMIE. The reason is that the DoKMIE are unlikely to list all possible instances of nonliteral language and because knowing that an expression *can* be used nonliterally does not mean that you can tell when it *is* being used nonliterally. The target verbs may not, and typically do not, appear in the feedback sets. In addition, the feedback sets are noisy and not annotated by any human, which is why we call TroFi unsupervised. When we use WordNet as a source of example sentences, or of seed words for pulling sentences out of the WSJ, for building the literal feedback set, we cannot tell if the WordNet synsets, or the collected feature sets, are actually literal. We provide some automatic methods in Section 3.3 to ensure that the feedback set feature sets that will harm us in the clustering phase are removed. As a side-effect, we may fill out sparse nonliteral sets.

In the next section we look at the Core TroFi algorithm and its use of the above data sources.

3.2 Core Algorithm

Since we are attempting to reduce the problem of literal/nonliteral recognition to one of word-sense disambiguation, TroFi makes use of an existing similarity-based word-sense disambiguation algorithm developed by (Karov & Edelman, 1998), henceforth KE.

The KE algorithm is based on the principle of attraction: similarities are calculated between sentences containing the word we wish to disambiguate (the *target word*) and collections of seed sentences (*feedback sets*) (see also Section 3.1).

A target set sentence is considered to be attracted to the feedback set containing the sentence to which it shows the highest similarity. Two sentences are similar if they contain similar words and two words are similar if they are contained in similar sentences. The resulting *transitive similarity* allows us to defeat the *knowledge acquisition bottleneck* – i.e. the low likelihood of finding all possible usages of a word in a single corpus. Note that the KE algorithm concentrates on similarities in the way sentences use the target literal or nonliteral word, not on similarities in the meanings of the sentences themselves.

Algorithms 1 and 2 summarize the basic TroFi version of the KE algorithm. Note that $p(w, s)$ is the unigram probability of word w in sentence s,

Algorithm 2 *KE-test*: classifying literal/nonliteral

1: For any sentence $s_x \in \mathcal{S}$
2: **if** $\max_{s_y}$ *s-sim*$^L(s_x, s_y) > \max_{s_y}$ *s-sim*$^N(s_x, s_y)$ **then**
3: tag s_x as literal
4: **else**
5: tag s_x as nonliteral
6: **end if**

normalized by the total number of words in s.

In practice, initializing *s-sim*$^I_0$ in line (2) of Algorithm 1 to 0 and then updating it from *w-sim*$_0$ means that each target sentence is still maximally similar to itself, but we also discover additional similarities between target sentences. We further enhance the algorithm by using Sum of Similarities. To implement this, in Algorithm 2 we change line (2) into: $\sum_{s_y}$ *s-sim*$^L(s_x, s_y) > \sum_{s_y}$ *s-sim*$^N(s_x, s_y)$

Although it is appropriate for fine-grained tasks like word-sense disambiguation to use the single highest similarity score in order to minimize noise, summing across all the similarities of a target set sentence to the feedback set sentences is more appropriate for literal/nonliteral clustering, where the usages could be spread across numerous sentences in the feedback sets. We make another modification to Algorithm 2 by checking that the maximum sentence similarity in line (2) is above a certain threshold for classification. If the similarity is above this threshold, we label a target-word sentence as literal or nonliteral.

Before continuing, let us look at an example. The features are shown in bold.

Target Set
1 The **girl** and her **brother** grasped their **mother**'s **hand**.
2 He thinks he has grasped the **essenti**als of the **institut**e's **finance philosophi**es.
3 The **presid**ent failed to grasp **ACTech**'s **finance quandary**.

Literal Feedback Set
L1 The **man**'s aging **mother** gripped her **husband**'s **shoulder**s tightly.
L2 The **child** gripped her **sister**'s **hand** to **cross** the **road**.
L3 The **presid**ent just doesn't get the picture, does he?

Nonliteral Feedback Set
N1 After much thought, he finally grasped the **idea**.
N2 This **idea** is risky, but it looks like the **director** of the **institut**e has comprehended the basic **principl**es behind it.
N3 Mrs. **Fipp**s is having trouble comprehending the legal **strait**s of the **institut**e.
N4 She had a **hand** in his fully comprehending the **quandary**.

The target set consists of sentences from the corpus containing the target word. The feedback sets contain sentences from the corpus containing synonyms of the target word found in WordNet (literal feedback set) and the DoKMIE (nonliteral feedback set). The feedback sets also contain example sentences provided in the target-word entries of these datasets. TroFi attempts to cluster the target set sentences into literal and nonliteral by attracting them to the corresponding feature sets using Algorithms 1 & 2. Using the basic KE algorithm, target sentence 2 is correctly attracted to the nonliteral set, and sentences 1 and 3 are equally attracted to both sets. When we apply our sum of similarities enhancement, sentence 1 is correctly attracted to the literal set, but sentence 3 is now incorrectly attracted to the literal set too. In the following sections we describe some enhancements – Learners & Voting, SuperTags, and Context – that try to solve the problem of incorrect attractions.

3.3 Cleaning the Feedback Sets

In this section we describe how we clean up the feedback sets to improve the performance of the Core algorithm. We also introduce the notion of Learners & Voting.

Recall that neither the raw data nor the collected feedback sets are manually annotated for training purposes. Since, in addition, the feedback sets are collected automatically, they are very noisy. For instance, in the example in Section 3.2, the literal feedback set sentence L3 contains an idiom which was provided as an example sentence in WordNet as a synonym for "grasp". In N4, we have the side-effect feature "hand", which unfortunately overlaps with the feature "hand" that we might hope to find in the literal set (e.g. "grasp his hand"). In order to remove sources of false attraction like these, we introduce the notion of *scrubbing*. Scrubbing is founded on a few basic principles. The first is that the contents of the DoKMIE come from (third-party) human annotations and are thus trusted. Consequently we take them as primary and use them to scrub the WordNet synsets. The second is that phrasal and expression verbs, for example "throw away", are often indicative of nonliteral uses of verbs – i.e. they are not the sum of their parts – so they can be used for scrubbing. The third is that content words appearing in both feedback sets – for example "the *wind* is blowing" vs. "the *winds* of war are blowing" for the target word "blow" – will lead to impure feedback sets, a situation we want to avoid. The fourth is that our scrubbing action can take a number of different forms: we can choose to scrub

just a word, a whole synset, or even an entire feature set. In addition, we can either move the offending item to the opposite feedback set or remove it altogether. Moving synsets or feature sets can add valuable content to one feedback set while removing noise from the other. However, it can also cause unforeseen contamination. We experimented with a number of these options to produce a whole complement of feedback set *learners* for classifying the target sentences. Ideally this will allow the different learners to correct each other.

For Learner A, we use *phrasal/expression verbs* and *overlap* as indicators to select whole WordNet *synsets* for *moving* over to the nonliteral feedback set. In our example, this causes L1-L3 to be moved to the nonliteral set. For Learner B, we use *phrasal/expression verbs* and *overlap* as indicators to *remove* problematic *synsets*. Thus we avoid accidentally contaminating the nonliteral set. However, we do end up throwing away information that could have been used to pad out sparse nonliteral sets. In our example, this causes L1-L3 to be dropped. For Learner C, we *remove feature sets* from the final literal and nonliteral feedback sets based on *overlapping words*. In our example, this causes L2 and N4 to be dropped. Learner D is the baseline – no scrubbing. We simply use the basic algorithm. Each learner has benefits and shortcomings. In order to maximize the former and minimize the latter, instead of choosing the single most successful learner, we introduce a *voting* system. We use a simple majority-rules algorithm, with the strongest learners weighted more heavily. In our experiments we double the weights of Learners A and D. In our example, this results in sentence 3 now being correctly attracted to the nonliteral set.

3.4 Additional Features

Even before voting, we attempt to improve the correctness of initial attractions through the use of SuperTags, which allows us to add internal structure information to the *bag-of-words* feature lists. *SuperTags* (Bangalore & Joshi, 1999) encode a great deal of syntactic information in a single tag (each tag is an elementary tree from the XTAG English Tree Adjoining Grammar). In addition to a word's part of speech, they also encode information about its location in a syntactic tree – i.e. we learn something about the surrounding words as well. We devised a SuperTag *trigram* composed of the SuperTag of the target word and the following two words and their SuperTags if they contain nouns, prepositions, particles, or adverbs. This is helpful in cases where the same set of features can be used as part of both literal and nonliteral expressions. For example, turning "It's hard to kick a habit like drinking" into "habit drink kick/B_nx0Vpls1_habit/A_NXN," results in a higher attraction to sentences about "kicking habits" than to sentences like "She has a habit of kicking me when she's been drinking."

Note that the creation of Learners A and B changes if SuperTags are used. In the original version, we only move or remove synsets based on phrasal/expression verbs and overlapping words. If SuperTags are used, we also move or remove feature sets whose SuperTag trigram indicates phrasal verbs (verb-particle expressions).

A final enhancement involves extending the context to help with disambiguation. Sometimes critical disambiguation features are contained not in the sentence with the target word, but in an adjacent sentence. To add context, we simply group the sentence containing the target word with a specified number of surrounding sentences and turn the whole group into a single feature set.

4 Results

TroFi was evaluated on the 25 target words listed in Table 1. The target sets contain from 1 to 115 manually annotated sentences for each verb. The first round of annotations was done by the first annotator. The second annotator was given no instructions besides a few examples of literal and nonliteral usage (not covering all target verbs). The authors of this paper were the annotators. Our inter-annotator agreement on the annotations used as test data in the experiments in this paper is quite high. κ (Cohen) *and* κ (S&C) on a random sample of 200 annotated examples annotated by two different annotators was found to be 0.77. As per ((Di Eugenio & Glass, 2004), cf. refs therein), the standard assessment for κ values is that tentative conclusions on agreement exists when $.67 \leq \kappa < .8$, and a definite conclusion on agreement exists when $\kappa \geq .8$.

In the case of a larger scale annotation effort, having the person leading the effort provide one or two examples of literal and nonliteral usages for each target verb to each annotator would almost certainly improve inter-annotator agreement. Table 1 lists the total number of target sentences, plus the manually evaluated literal and nonliteral

counts, for each target word. It also provides the feedback set sizes for each target word. The totals across all words are given at the bottom of the table.

	absorb	assault	die	drag	drown
Lit Target	4	3	24	12	4
Nonlit Target	62	0	11	41	1
Target	66	3	35	53	5
Lit FB	286	119	315	118	25
Nonlit FB	1	0	7	241	21
	escape	examine	fill	fix	flow
Lit Target	24	49	47	39	10
Nonlit Target	39	37	40	16	31
Target	63	86	87	55	41
Lit FB	124	371	244	953	74
Nonlit FB	2	2	66	279	2
	grab	grasp	kick	knock	lend
Lit Target	5	1	10	11	77
Nonlit Target	13	4	26	29	15
Target	18	5	36	40	92
Lit FB	76	36	19	60	641
Nonlit FB	58	2	172	720	1
	miss	pass	rest	ride	roll
Lit Target	58	0	8	22	25
Nonlit Target	40	1	20	26	46
Target	98	1	28	48	71
Lit FB	236	1443	42	221	132
Nonlit FB	13	156	6	8	74
	smooth	step	stick	strike	touch
Lit Target	0	12	8	51	13
Nonlit Target	11	94	73	64	41
Target	11	106	81	115	54
Lit FB	28	5	132	693	904
Nonlit FB	75	517	546	351	406
Totals: Target=1298; Lit FB=7297; Nonlit FB=3726					

Table 1: Target and Feedback Set Sizes.

The algorithms were evaluated based on how accurately they clustered the hand-annotated sentences. Sentences that were attracted to neither cluster or were equally attracted to both were put in the opposite set from their label, making a failure to cluster a sentence an incorrect clustering.

Evaluation results were recorded as *recall*, *precision*, and *f-score* values. *Literal recall* is defined as *(correct literals in literal cluster / total correct literals)*. *Literal precision* is defined as *(correct literals in literal cluster / size of literal cluster)*. If there are no literals, *literal recall* is 100%; *literal precision* is 100% if there are no nonliterals in the literal cluster and 0% otherwise. The *f-score* is defined as *(2 · precision · recall) / (precision + recall)*. Nonliteral precision and recall are defined similarly. Average precision is the average of literal and nonliteral precision; similarly for average recall. For overall performance, we take the f-score of average precision and average recall.

We calculated two baselines for each word. The first was a simple majority-rules baseline. Due to the imbalance of literal and nonliteral examples, this baseline ranges from 60.9% to 66.7% with an average of 63.6%. Keep in mind though that using this baseline, the f-score for the nonliteral set will always be 0%. We come back to this point at the end of this section. We calculated a second baseline using a simple attraction algorithm. Each target set sentence is attracted to the feedback set containing the sentence with which it has the most words in common. This corresponds well to the basic *highest similarity* TroFi algorithm. Sentences attracted to neither, or equally to both, sets are put in the opposite cluster to where they belong. Since this baseline actually attempts to distinguish between literal and nonliteral *and* uses all the data used by the TroFi algorithm, it is the one we will refer to in our discussion below.

Experiments were conducted to first find the results of the core algorithm and then determine the effects of each enhancement. The results are shown in Figure 1. The last column in the graph shows the average across all the target verbs.

On average, the basic TroFi algorithm (KE) gives a 7.6% improvement over the baseline, with some words, like “lend” and “touch”, having higher results due to transitivity of similarity. For our sum of similarities enhancement, all the individual target word results except for “examine” sit above the baseline. The dip is due to the fact that while TroFi can generate some beneficial similarities between words related by context, it can also generate some detrimental ones. When we use sum of similarities, it is possible for the transitively discovered indirect similarities between a target nonliteral sentence and all the sentences in a feedback set to add up to more than a single direct similarity between the target sentence and a single feedback set sentence. This is not possible with highest similarity because a single sentence would have to show a higher similarity to the target sentence than that produced by sharing an identical word, which is unlikely since transitively discovered similarities generally do not add up to 1. So, although highest similarity occasionally produces better results than using sum of similarities, on average we can expect to get better results with the latter. In this experiment alone, we get an average f-score of 46.3% for the sum of similarities results – a 9.4% improvement over the high similarity results (36.9%) and a 16.9% improvement over the baseline (29.4%).

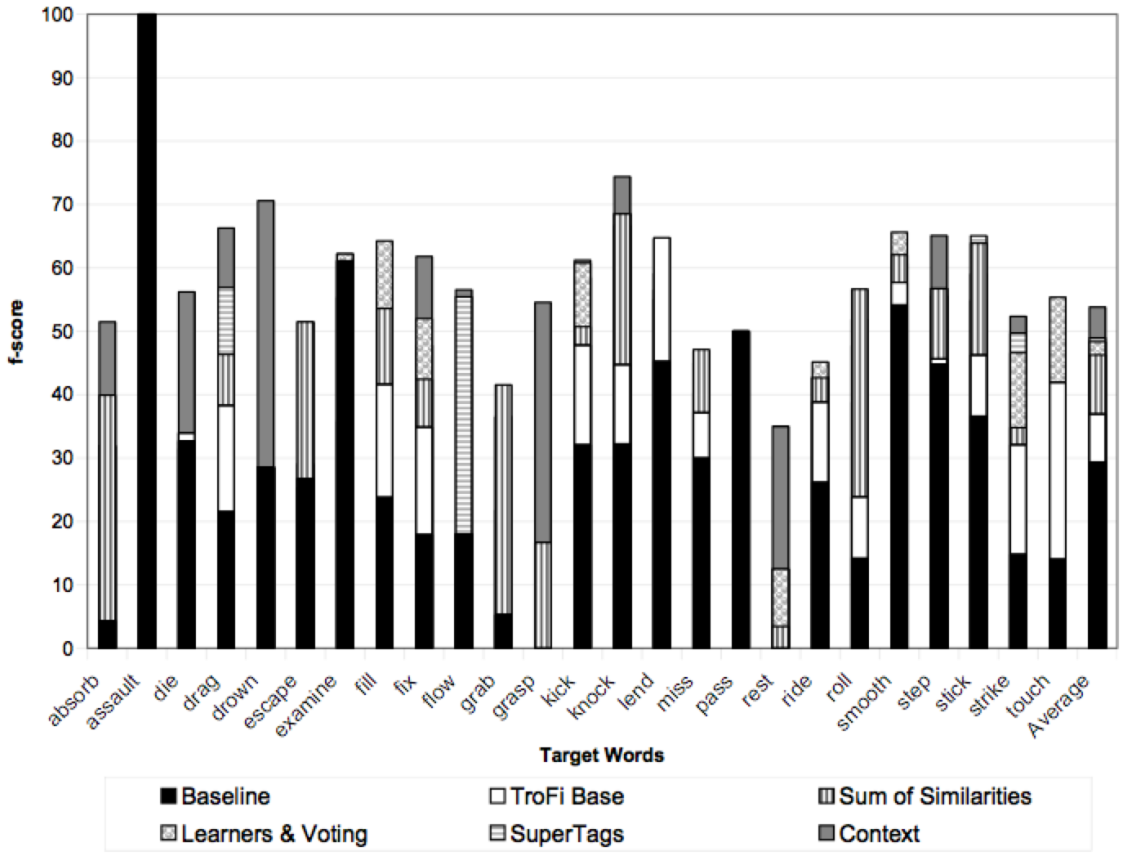

Figure 1: TroFi Evaluation Results.

In comparing the individual results of all our learners, we found that the results for Learners A and D (46.7% and 46.3%) eclipsed Learners B and C by just over 2.5%. Using majority-rules voting with Learners A and D doubled, we were able to obtain an average f-score of 48.4%, showing that voting does to an extent balance out the learners' varying results on different words.

The addition of SuperTags caused improvements in some words like "drag" and "stick". The overall gain was only 0.5%, likely due to an overgeneration of similarities. Future work may identify ways to use SuperTags more effectively.

The use of additional context was responsible for our second largest leap in performance after sum of similarities. We gained 4.9%, bringing us to an average f-score of 53.8%. Worth noting is that the target words exhibiting the most significant improvement, "drown" and "grasp", had some of the smallest target and feedback set feature sets, supporting the theory that adding cogent features may improve performance.

With an average of 53.8%, all words but one lie well above our simple-attraction baseline, and some even achieve much higher results than the majority-rules baseline. Note also that, using this latter baseline, TroFi boosts the nonliteral f-score from 0% to 42.3%.

5 The TroFi Example Base

In this section we discuss the TroFi Example Base. First, we examine *iterative augmentation*. Then we discuss the structure and contents of the example base and the potential for expansion.

After an initial run for a particular target word, we have the cluster results plus a record of the feedback sets augmented with the newly clustered sentences. Each feedback set sentence is saved with a *classifier weight*, with newly clustered sentences receiving a weight of 1.0. Subsequent runs may be done to augment the initial clusters. For these runs, we use the *classifiers* from our initial run as feedback sets. New sentences for clustering are treated like a regular target set. Running TroFi produces new clusters and re-weighted classifiers augmented with newly clustered sentences. There can be as many runs as desired; hence *iterative augmentation*.

We used the iterative augmentation process to build a small example base consisting of the target words from Table 1, as well as another 25 words drawn from the examples of scholars whose work

*****pour*****
nonliteral cluster
wsj04:7878 N As manufacturers get bigger , they are likely to pour more money into the battle for shelf space , raising the ante for new players ./.
wsj25:3283 N Salsa and rap music pour out of the windows ./.
wsj06:300 U Investors hungering for safety and high yields are pouring record sums into single-premium , interest-earning annuities ./.
literal cluster
wsj59:3286 L Custom demands that cognac be poured from a freshly opened bottle ./.

Figure 2: TroFi Example Base Excerpt.

was reviewed in Section 2. It is important to note that in building the example base, we used TroFi with an Active Learning component (see (Birke, 2005)) which improved our average f-score from 53.8% to 64.9% on the original 25 target words.

An excerpt from the example base is shown in Figure 2. Each entry includes an ID number and a Nonliteral, Literal, or Unannotated tag. Annotations are from testing or from active learning during example-base construction. The TroFi Example Base is available at *http://www.cs.sfu.ca/~anoop/students/jbirke/*. Further unsupervised expansion of the existing clusters as well as the production of additional clusters is a possibility.

6 Conclusion

In this paper we presented TroFi, a system for separating literal and nonliteral usages of verbs through statistical word-sense disambiguation and clustering techniques. We suggest that TroFi is applicable to all sorts of *nonliteral* language, and that, although it is currently focused on English *verbs*, it could be adapted to other parts of speech and other languages.

We adapted an existing word-sense disambiguation algorithm to literal/nonliteral clustering through the redefinition of literal and nonliteral as word senses, the alteration of the similarity scores used, and the addition of learners and voting, SuperTags, and additional context.

For all our models and algorithms, we carried out detailed experiments on hand-annotated data, both to fully evaluate the system and to arrive at an optimal configuration. Through our enhancements we were able to produce results that are, on average, 16.9% higher than the core algorithm and 24.4% higher than the baseline.

Finally, we used our optimal configuration of TroFi, together with active learning and iterative augmentation, to build the TroFi Example Base, a publicly available, expandable resource of literal/nonliteral usage clusters that we hope will be useful not only for future research in the field of nonliteral language processing, but also as training data for other statistical NLP tasks.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Comput. Linguist.* 25, 2 (Jun. 1999), 237-265.

Julia Birke. 2005. *A Clustering Approach for the Unsupervised Recognition of Nonliteral Language*. M.Sc. Thesis. School of Computing Science, Simon Fraser University.

Barbara Di Eugenio and Michael Glass. 2004. The kappa statistic: a second look. *Comput. Linguist.* 30, 1 (Mar. 2004), 95-101.

William B. Dolan. 1995. Metaphor as an emergent property of machine-readable dictionaries. In Proceedings of Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity (March 1995, Stanford University, CA). AAAI 1995 Spring Symposium Series, 27-29.

Dan Fass. 1997. *Processing metonymy and metaphor*. Greenwich, CT: Ablex Publishing Corporation.

Yael Karov and Shimon Edelman. 1998. Similarity-based word sense disambiguation. *Comput. Linguist.* 24, 1 (Mar. 1998), 41-59.

James H. Martin. 1990. *A computational model of metaphor interpretation*. Toronto, ON: Academic Press, Inc.

James H. Martin. 1992. Computer understanding of conventional metaphoric language. *Cognitive Science* 16, 2 (1992), 233-270.

Zachary J. Mason. 2004. CorMet: a computational, corpus-based conventional metaphor extraction system. *Comput. Linguist.* 30, 1 (Mar. 2004), 23-44.

Masaki Murata, Qing Ma, Atsumu Yamamoto, and Hitoshi Isahara. 2000. Metonymy interpretation using *x no y* examples. In *Proceedings of SNLP2000* (Chiang Mai, Thailand, 10 May 2000).

Srini Narayanan. 1999. Moving right along: a computational model of metaphoric reasoning about events. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th IAAI Conference* (Orlando, US, 1999). 121-127.

Malvina Nissim and Katja Markert. 2003. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (ACL-03) (Sapporo, Japan, 2003). 56-63.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference* (University of Pennsylvania, May 17-18 1996).

Sylvia W. Russell. 1976. Computer understanding of metaphorically used verbs. *American Journal of Computational Linguistics*, Microfiche 44.