Textual Inference for Machine Comprehension

Martin Gleize

NNT : 2016SACLS004

THÈSE DE DOCTORAT
DE
L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À
L'UNIVERSITÉ PARIS-SUD

ECOLE DOCTORALE N° 580
Sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat : Informatique

Par

M. Martin Gleize

Textual Inference for Machine Comprehension

Thèse présentée et soutenue à Orsay, le 7 janvier 2016

Composition du Jury :

M. Yvon François	Professeur, Université Paris-Sud	Président, Examinateur
Mme Gardent Claire	Directeur de recherche CNRS, LORIA	Rapporteur
M. Magnini Bernardo	Senior Researcher, FBK	Rapporteur
M. Piwowarski Benjamin	Chargé de recherche CNRS, LIP6	Examinateur
Mme Grau Brigitte	Professeur, ENSIIE	Directeur de thèse

Abstract

With the ever-growing mass of published text, natural language understanding stands as one of the most sought-after goal of artificial intelligence. In natural language, not every fact expressed in the text is necessarily written: human readers naturally infer what is missing through various intuitive linguistic skills, common sense or domain-specific knowledge, and life experiences. Natural Language Processing (NLP) systems do not have these initial capabilities. Unable to draw inferences to fill the gaps in the text, they cannot truly understand it. This dissertation focuses on this problem and presents our work on the automatic resolution of textual inferences in the context of machine reading.

A textual inference is simply defined as a relation between two fragments of text: a human reading the first can reasonably infer that the second is true. A lot of different NLP tasks more or less directly evaluate systems on their ability to recognize textual inference. Among this multiplicity of evaluation frameworks, inferences themselves are not one and the same and also present a wide variety of different types. We reflect on inferences for NLP from a theoretical standpoint and present two contributions addressing these levels of diversity: an abstract contextualized inference task encompassing most NLP inference-related tasks, and a novel hierchical taxonomy of textual inferences based on their difficulty.

Automatically recognizing textual inference currently almost always involves a machine learning model, trained to use various linguistic features on a labeled dataset of samples of textual inference. However, specific data on complex inference phenomena is not currently abundant enough that systems can directly learn world knowledge and commonsense reasoning. Instead, systems focus on learning how to use the syntactic structure of sentences to align the words of two semantically related sentences. To extend what systems know of the world, they include external background knowledge, often improving their results. But this addition is often made on top of other features, and rarely well integrated to sentence structure.

The main contributions of our thesis address the previous concern, with the aim of solving complex natural language understanding tasks. With the hypothesis that a simpler lexicon should make easier to compare the sense of two sentences, we present a passage retrieval method using structured lexical expansion backed up by a simplifying dictionary. This simplification hypothesis is tested again in a contribution on textual entailment: syntactical paraphrases are extracted from the

same dictionary and repeatedly applied on the Text to turn it into the Hypothesis. We then present a machine learning kernel-based method recognizing sentence rewritings, with a notion of types able to encode lexical-semantic knowledge. This approach is effective on three tasks: paraphrase identification, textual entailment and question answering. We address its lack of scalability while keeping most of its strengths in our last contribution. Reading comprehension tests are used for evaluation: these multiple-choice questions on short text constitute the most practical way to assess textual inference within a complete context. Our system is founded on a efficient tree edit algorithm, and the features extracted from edit sequences are used to build two classifiers for the validation and invalidation of answer candidates. This approach reaches second place at the "Entrance Exams" CLEF 2015 challenge.

Résumé

Etant donnée la masse toujours croissante de texte publié, la compréhension automatique des langues naturelles est à présent l'un des principaux enjeux de l'intelligence artificielle. En langue naturelle, les faits exprimés dans le texte ne sont pas nécessairement tous explicites : le lecteur humain infère les éléments manquants grâce à ses compétences linguistiques, ses connaissances de sens commun ou sur un domaine spécifique, et son expérience. Les systèmes de Traitement Automatique des Langues (TAL) ne possèdent naturellement pas ces capacités. Incapables de combler les défauts d'information du texte, ils ne peuvent donc pas le comprendre vraiment. Cette thèse porte sur ce problème et présente notre travail sur la résolution d'inférences pour la compréhension automatique de texte.

Une inférence textuelle est définie comme une relation entre deux fragments de texte : un humain lisant le premier peut raisonnablement inférer que le second est vrai. Beaucoup de tâches de TAL évaluent plus ou moins directement la capacité des systèmes à reconnaître l'inférence textuelle. Au sein de cette multiplicité de l'évaluation, les inférences elles-mêmes présentent une grande variété de types. Nous nous interrogeons sur les inférences en TAL d'un point de vue théorique et présentons deux contributions répondant à ces niveaux de diversité : une tâche abstraite contextualisée qui englobe les tâches d'inférence du TAL, et une taxonomie hiérarchique des inférences textuelles en fonction de leur difficulté.

La reconnaissance automatique d'inférence textuelle repose aujourd'hui presque toujours sur un modèle d'apprentissage, entraîné à l'usage de traits linguistiques variés sur un jeu d'inférences textuelles étiquetées. Cependant, les données spécifiques aux phénomènes d'inférence complexes ne sont pour le moment pas assez abondantes pour espérer apprendre automatiquement la connaissance du monde et le raisonnement de sens commun nécessaires. Les systèmes actuels se concentrent plutôt sur l'apprentissage d'alignements entre les mots de phrases reliées sémantiquement, souvent en utilisant leur structure syntaxique. Pour étendre leur connaissance du monde, ils incluent des connaissances tirées de ressources externes, ce qui améliore souvent les performances. Mais cette connaissance est souvent ajoutée par dessus les fonctionnalités existantes, et rarement bien intégrée à la structure de la phrase.

Nos principales contributions dans cette thèse répondent au problème précédent. En partant de l'hypothèse qu'un lexique plus simple devrait rendre plus facile la comparaison du sens de deux phrases, nous décrivons une méthode de récupération de passage fondée sur une expansion lexicale structurée et un dictionnaire de simplifications. Cette hypothèse est testée à nouveau dans une de nos

contributions sur la reconnaissance d'implication textuelle : des paraphrases syntaxiques sont extraites du dictionnaire et appliquées récursivement sur la première phrase pour la transformer en la seconde. Nous présentons ensuite une méthode d'apprentissage par noyaux de réécriture de phrases, avec une notion de types permettant d'encoder des connaissances lexico-sémantiques. Cette approche est efficace sur trois tâches : la reconnaissance de paraphrases, d'implication textuelle, et le question-réponses. Nous résolvons son problème de passage à l'échelle dans une dernière contribution. Des tests de compréhension sont utilisés pour son évaluation, sous la forme de questions à choix multiples sur des textes courts, qui permettent de tester la résolution d'inférences en contexte. Notre système est fondé sur un algorithme efficace d'édition d'arbres, et les traits extraits des séquences d'édition sont utilisés pour construire deux classifieurs pour la validation et l'invalidation des choix de réponses. Cette approche a obtenu la deuxième place du challenge "Entrance Exams" à CLEF 2015.

Synthèse en français

Etant donnée la masse toujours croissante de texte publié, la comprehension automatique des langues naturelles est à présent l'un des principaux enjeux de l'intelligence artificielle. En langue naturelle, les faits exprimés dans le texte ne sont pas nécessairement tous explicites : le lecteur humain infère les éléments manquants grâce à ses compétences linguistiques, ses connaissances de sens commun ou sur un domaine spécifique, et son expérience. Les systèmes de Traitement Automatique des Langues (TAL) ne possèdent naturellement pas ces capacités. Incapables de combler les défauts d'information du texte, ils ne peuvent donc pas le comprendre vraiment. Cette thèse porte sur ce problème et présente notre travail sur la résolution d'inférences pour la compréhension automatique de texte.

Une inférence textuelle est définie comme une relation directionnelle entre deux fragments de texte, le texte T et l'hypothèse H : un humain lisant T peut raisonnablement inférer que H est vraie (Dagan et al., 2006). Bien que le terme *implication textuelle* désigne généralement la même relation, nous le réservons pour la tâche spécifique d'évaluation nommée *Recognizing Textual Entailment*, introduite par Dagan et al. (2006). Ci-dessous se trouve un exemple d'inférence textuelle :

(1) **T:** *The sun was still high in the sky, but John was already getting sleepy.*
H: *John is awake.*

John est somnolent, sur le point de s'endormir, ce qui prouve qu'il n'est justement pas encore en train de dormir. Cet exemple serait une instance positive dans la tâche RTE, qui consiste à classifier ces couples de phrases en couples d'inférence textuelle et couples ne représentant pas la relation.

Le processus d'inférence est l'un des moteurs de la compréhension de texte chez les êtres humains. Le texte T dans l'exemple précédent apporte de la connaissance implicite sur John : ce n'est apparemment pas normal que John veuille dormir en pleine journée. Cette inférence est déjà complexe, mais elle déclenche de multiples inférences potentielles, comme la raison derrière l'état de John. Si le texte T n'était qu'une partie d'un plus long texte, nous pourrions nous attendre à découvrir cette raison avant ou après cette phrase, et peut-être même que nous la rechercherions activement. Ce type de raisonnement demeure toujours à l'esprit d'un lecteur humain compétent, mais est pour l'instant hors de portée des ordinateurs, ce qui fait de la compréhension automatique un problème encore largement ouvert.

Beaucoup de tâches de TAL évaluent plus ou moins directement la capacité des systèmes à reconnaître l'inférence textuelle. Au sein de cette multiplicité de l'évaluation, les inférences elles-mêmes présentent une grande variété de types. Nous nous intérrogeons sur les inférences en TAL d'un point de vue théorique et présentons deux contributions répondant à ces niveaux de diversité : une tâche abstraite contextualisée qui englobe les tâches d'inférence du TAL, et une taxonomie hiérarchique des inférences textuelles en fonction de leur difficulté.

La reconnaissance automatique d'inférence textuelle repose aujourd'hui presque toujours sur un modèle d'apprentissage, entraîné à l'usage de traits linguistiques variés sur un jeu d'inférences textuelles étiquetées. Cependant, les données spécifiques aux phénomènes d'inférence complexes ne sont pour le moment pas assez abondantes pour espérer apprendre automatiquement la connaissance du monde et le raisonnement de sens commun nécessaires. Les systèmes actuels se concentrent plutôt sur l'apprentissage d'alignements entre les mots de phrases reliées sémantiquement, souvent en utilisant leur structure syntaxique. Pour étendre leur connaissance du monde, ils incluent des connaissances tirées de ressources externes, ce qui améliore souvent les performances. Mais cette connaissance est souvent ajoutée par dessus les fonctionalités existantes, et rarement bien intégrée à la structure de la phrase.

Nos principales contributions dans cette thèse répondent au problème précédent. En partant de l'hypothèse qu'un lexique plus simple devrait rendre plus facile la comparaison du sens de deux phrases, nous décrivons une méthode de récupération de passage fondée sur une expansion lexicale structurée et un dictionnaire de simplifications. Cette hypothèse est testée à nouveau dans une de nos contributions sur la reconnaissance d'implication textuelle : des paraphrases syntaxiques sont extraites du dictionnaire et appliquées récursivement sur la première phrase pour la transformer en la seconde. Nous présentons ensuite une méthode d'apprentissage par noyaux de réécriture de phrases, avec une notion de types permettant d'encoder des connaissances lexico-sémantiques. Cette approche est efficace sur trois tâches : la reconnaissance de paraphrases, d'implication textuelle, et le question-réponses. Nous résolvons son problème de passage à l'échelle dans une dernière contribution. Des tests de compréhension sont utilisés pour son évaluation, sous la forme de questions à choix multiples sur des textes courts, qui permettent de tester la résolution d'inférences en contexte. Notre système est fondé sur un algorithme efficace d'édition d'arbres, et les traits extraits des séquences d'édition sont utilisés pour construire deux classifieurs pour la validation et l'invalidation des choix de réponses. Cette approche a obtenu la deuxième

place du challenge "Entrance Exams" à CLEF 2015.

Contents

Introduction

With the ever-growing mass of published text, natural language understanding stands as one of the most sought-after goal of artificial intelligence (AI). One of its research subtopics, machine reading, consists in building a machine, or an algorithm, capable of automatically reading a text written in natural language and understanding it, usually so as to represent and extract knowledge in a structured way for further use in other AI tasks.

One of the characteristics of language is that not every fact expressed in the text is necessarily written: human readers naturally infer what is missing through various intuitive linguistic skills, common sense or domain-specific knowledge, and life experiences. Machines typically do not have these initial capabilities. Unable to draw inferences to fill the gaps in the text, they cannot truly understand it. This dissertation focuses on this problem and presents our work on the automatic resolution of textual inferences in the context of machine reading.

Textual inference is defined as a directional relation between two text fragments T, for *text*, and H, *for hypothesis*. The statements "H can be inferred from T" or "T entails H" both mean that a human reading T can reasonably infer that H is true (Dagan et al., 2006). Although the term *textual entailment* generally designates the same relation, we will reserve this phrase for the specific evaluation task of Recognizing Textual Entailment (RTE), as introduced by Dagan et al. (2006). Example 1 shows such a textual inference:

(2) **T:** *The sun was still high in the sky, but John was already getting sleepy.*
H: *John is awake.*

John is getting sleepy, which proves that he is not sleeping yet. This example would thus be a positive instance of the RTE task, which consists in classifying those pairs of sentences according to whether they constitute an entailment relation or not.

The process of making inferences is one of the engines that drive reading

comprehension for human beings. The text T in the previous example carries implicit knowledge about John: it is apparently not normal for John to want to sleep while it is still daytime. This is already a complex inference, but it triggers more potential inferences to make, like the reason for John to be in such a state. If text T was part of a longer text, we would expect the reason to be explained at some point before or after this sentence and maybe even start to look for it actively. This kind of reasoning is always in the mind of a competent human reader, but is still as of now inaccessible to computers, making machine reading an unsolved problem.

Inferences are not one and the same however, so we can expect our systems to be able to solve the easier ones, or inferences of a particular type, and then decide from there in which direction to make improvements. The current literature only scarcely describes ways to classify inference phenomena in useful ways for Natural Language Processing (NLP). Our first work in this thesis was to reflect on inferences for NLP from a theoretical standpoint. This led us to two contributions. The first is a novel hierarchy of textual inferences based on their difficulty. The second is the design of a preliminary formal system to solve most NLP high-level tasks based on textual inference in a unified manner.

Question Answering (QA) is a long-standing problem and a well-studied task in its factoid version, but complex questions needing inference were rarely the center of attention until more recently. With the appearance of the RTE evaluation campaigns in 2005 (Dagan et al., 2006), the research community could distance itself from information retrieval and focus more keenly on the sole task of recognizing textual entailment; most of the present and past techniques that deal directly with textual inferences are thus evaluated on the RTE datasets. The current trend almost always involves a machine learning model, trained on a labeled dataset to use various linguistic features.

However, specific data on complex questions or entailment phenomena are not currently abundant enough that systems can learn world knowledge and commonsense reasoning from them. Instead, systems which attempt to solve those complex tasks focus on learning how to use the syntactic structures of sentences or what to make of pre-defined word alignment scoring functions. Including external background knowledge to extend what the system knows of the world is an intuitive idea, and is often attempted in practice, with decent results. But this inclusion is often made on top of other features of the system, and rarely well integrated to sentence structure.

The main contributions of our thesis address this concern, with the aim of

solving complex natural language understanding tasks. With the hypothesis that a simpler lexicon should make easier to compare the sense of two sentences, we present a passage retrieval method using structured lexical expansion backed up by a simplifying dictionary. This simplification hypothesis is tested again in a contribution on textual entailment: syntactical paraphrases are extracted from the same dictionary and repeatedly applied on the Text to turn it into the Hypothesis. We then present a machine learning kernel-based method recognizing sentence rewritings, with a notion of types able to encode lexical-semantic knowledge. This approach is effective on three tasks: paraphrase identification, textual entailment and question answering. We address its lack of scalability while keeping most of its strengths in our last contribution evaluated on reading comprehension tests. This complete system is founded on a efficient tree edit algorithm, and the features extracted from edit sequences are used to build two classifiers for the validation and invalidation of answer candidates.

With the knowledge that inferences are manifold, systems can be progressively designed to take on increasingly harder inferences, but on the practical end, this raises the question of how to evaluate progress on machine reading. An idea for an intrinsic evaluation would be to list all the interesting facts that a human being can retrieve in a text, and treat machine reading as an information retrieval task. Due to the infinite amount of valid conclusions to draw, this is not done in practice, especially on open-domain texts. However, there is an extrinsic way to evaluate a system on its reading comprehension: asking it questions about the text, like we would do for human readers. Reading comprehension tests are often composed of multiple-choice questions (MCQ). They are an especially good evaluation method both of human and machine readers, as they are easy to grade due to the constrained way the answer is formulated. As a consequence, building systems which automatically answer these reading comprehension tests was the main applicative direction of this thesis.

This dissertation is composed of six chapters. The first chapter presents the reflection which motivated our thesis, from the choice of subject to the direction of our research. The second chapter gives a review of the literature on textual inference up to the middle of 2015. In this part, we first describe the major related tasks that are currently tackled by the NLP community and their evaluation methods. We then address the systems in term of the increasingly complex linguistic elements they leverage, and the methods they implement to use these features of the text.

The third chapter presents our novel formal system designed to solve most NLP high-level tasks based on textual inference in a unified manner. This system is the answer to our intention of designing theoretical systems that are robust, extensible and mindful of most obstacles met when dealing with textual inferences.

The subsequent chapters each deal with contributions to solve high-level evaluation tasks. Each was evaluated on standard datasets and validated by being presented at an international conference. The core material of those chapters can be found individually in our publications. Chapter 4 presents systems which use the hypothesis that simplifying the lexicon should simplify the process of detecting semantic equivalences. By replacing words with their definition in a simplifying dictionary like the Simple English Wiktionary, we hope to be able to bring closer the ones with similar senses. Chapter 5 frames identifying inferences as being able to tell if a sentence rewrites into another. We propose a kernel method to detect sentence rewritings, and a tree-edit method, which are both able to use external knowledge sources to help with the rewriting process. Chapter 6 presents our work on automatically answering reading comprehension tests in multiple-choice question form. This problem remained the main underlying goal of our thesis throughout and each of our three years was rhythmically punctuated by a participation at a CLEF evaluation workshop on automatically solving reading comprehension tests for humans. In this last part, we describe this task, our study of the corpus, and our systems, with performance and error analysis.

Chapter 1

The need for inference

Reasoning is a necessary and natural activity of the human mind. Our environment provides us with evidence, we use it to reach conclusions, and these inferences help us make decisions and build complicated ideas. As an increasingly prominent part of our environment, textual data contains a wealth of useful information, but its sheer mass makes it hard to use effectively without help from computers.

In this chapter, we introduce what motivates research on automatic textual inference. We first explain the reasons behind inference, from human cognition to specific work in natural language processing. After presenting what the main inference-related NLP tasks are, we express them all in a single general problem. We finally propose a hierarchy of inference classes for NLP, to better exhibit what is possible today and what is missing for tomorrow.

1.1 What is a textual inference?

1.1.1 In human cognition

It is important to realize that there is no particular clever reason to model machine inference after human inference. After all, machines and humans are of different nature, and their relation to textual data is different too. We ask of machines what we cannot do ourselves: to handle the ever-growing mass of textual information rapidly. We do not require them to do it just like faster humans, or like many humans with a lot of time on their hands. But these texts are produced by human minds, and somewhere along the way, a machine encounters a gap between two sentences, a missing piece, something that a human mind instantly and naturally provides but the machine fails to. In fact, the machine cannot detect that the puzzle is not complete and will likely carry this erroneous understanding at all times, unsuspecting, while executing its task. We do not currently have any other insight on reading comprehension than what we know or suspect of ourselves. Therefore, looking at textual inferences in human cognition appears to be a fairly good starting point.

1.1.1.1 Parallels between human and machine readers

Research in Artificial Intelligence on textual inference can be traced back to the work of Schank et al. (1973) on single-sentence inference. Several years later, Norvig (1987) presents the reader of a text as faced with a formidable task: recognizing the individual words of the text, deciding how they are structured into sentences, determining the explicit meaning of each sentence, and also making *inferences* about the likely implicit meaning of each sentence, and the implicit connections between sentences. An inference is defined to be any assertion which the reader comes to believe to be true as a result of reading the text, but which was not previously believed by the reader, and was not stated explicitly in the text. Inferences do not need to follow logically from the text; the reader can jump to conclusions that seem likely but are not 100% certain. This is a definition that the natural language processing community mostly retained, with an obvious emphasis on inferences being in the end consistent in their conclusions with the supporting facts they are supplied.

Reading comprehension is an essential skill required of us early on in our life, and whose incomplete acquisition has heavy ramifications as we grow; it is natural that many works in the fields of psychology, education, and cognitive sciences

focus on young children's reading comprehension. This is interesting to the NLP community in that computers can somewhat be compared to young children in respect to the task of reading a text: they have to be taught from the ground up. Young children's reading comprehension skill has been associated in early studies with their ability to draw inferences (Oakhill, 1982). Despite intensive instruction, many children and adolescents fail to reach functional levels of reading comprehension, and the literature abundantly focuses on why. Although various theoretical models emphasize different aspects of reading comprehension, they share the central notion that, at its core, reading comprehension involves the construction of a coherent mental representation of the text in the readers' memory (Kendeou et al., 2014). This mental representation of the text includes textual information and associated background knowledge interconnected via semantic relations. Semantic relations are identified by the reader through passive and strategic inferential processes. The passive inferential processes take place automatically but the strategic processes demand readers' attentional and working memory resources. We argue that some major problems that children face when reading a text do not exist for computers, or at least are not to be urgently addressed. Difficulties related to attention allocation and working or long-term memories – their size in particular – are of little concern for today's computing systems. Computers are single-minded and hard-working in essence, and their memory is always increasing. There is no point in distinguishing passive and strategic inferences in terms of the effort computers have to devote to draw them. However, most problems that children encounter, computers do as well: word decoding, vocabulary knowledge for low level processes, semantic representation building, background world knowledge and inference making for high level processes (Cain et al., 2001).

1.1.1.2 Types of inference

Cognitive psychology identifies several types of inferences for reading comprehension. The most useful distinction for this thesis is between bridging and elaborative inferences. Bridging inferences, also called coherence inferences, maintain textual integrity (Kispal, 2008). For example, let us consider the following sentence:

(3) Peter begged his mother to let him go to the party.

The reader would have to realize that the pronouns 'his' and 'him' refer to Peter to fully understand the meaning. What is usually called coreference resolution in the field of NLP is not the only phenomenon encompassed by bridging inferences.

At its core, these inferences are essentially about identifying textual expressions that are semantically equivalent even if they use different words.

Elaborative inferences, on the other side, are not required for textual coherence and even basic comprehension. They serve to enrich the mental representation of the text and therefore make it more memorable. Generally knowledge-based, they include inferences about the consequences of an action, predictions about forthcoming events, speculations regarding the instrument used to perform an action and suppositions about the physical properties of characters and objects. By way of illustration, consider the following passage:

(4) The knight lunged towards the dragon and pierced his shining scales. The dragon turned towards the knight and let out a fiery roar.

It is possible to make many elaborations from this short passage. It could be inferred that the knight attacked the dragon with a sword because that is the usual tool of the knight and is implied by the verb 'pierced'. However, this is not necessarily the case since the knight could have been using any other sharp object had he not been in possession of his sword. Furthermore, it could be assumed that the knight was wounded by the dragon's fiery breath. However, it may be the case that the knight was able to avoid the dragon's attack. Although these elaborations enrich the model of the text, none of them is necessary in order to form a full and intelligible representation.

We argue that these two types of inference can overlap and interleave so that the distinction loses some of its potency. The following sentence is presented as an example of text prompting an elaborative inference (Kispal, 2008).

(5) Katy dropped the vase. She ran for the dustpan and brush to sweep up the pieces.

Indeed, the reader would have to draw upon life experience and general knowledge to realize that the vase broke, something of which the text does not make mention. It enriches the mental representation of the text. However, one can argue the necessity of this inference: it seems that this inference enriches this mental representation *in order to* build a coherent model and link the "pieces" to the "vase", hence making it a bridging reference too. Without it, the reader would be left wondering what happened to the vase and what those pieces are that popped in suddenly. We believe that the simpler first distinction for computers to make about inferences is between necessary and unnecessary inferences. And of course, it seems natural that NLP concerns itself primarily with the necessary

kind, but it is important to realize that what may be an unnecessary inference at the first reading of a text may become necessary when asked a question about it. Hence, there is interest in automatically drawing all kinds of inferences on texts.

1.1.2 In Natural Language Processing

At its simplest, a textual inference is an assertion or a piece of knowledge that can be *reasonably* concluded from textual premises. The premises are deemed true and complete, and tasks which evaluate inferring skills are more interested in the correctness of the process rather than the absolute truth of the conclusion as a self-standing fact. More interestingly, the word "reasonable" opens up a vast avenue of conflicting interpretations. It mainly means that we do not require that the conclusion is completely supported by all the facts, but commonsense may be used to add "everyday life" hypotheses to the problem. The two following examples, drawn from RTE3's test data (Giampiccolo et al., 2007), demonstrate the kind of assumptions that might be needed.

(6) **T:** *Rival generic drug maker ourlan Laboratories Inc. reported revenue of $1.3 billion for fiscal 2005.*
H: *ourlan Laboratories Inc. earned $1.3 billion in fiscal year 2005.*

(7) **T:** *Ms. Minton left Australia in 1961 to pursue her studies in London.*
H: *Ms. Minton was born in Australia.*

The example 6 is a textual entailment, but the example 7 is not. It is assumed that generally, if this only thing said of a company is that it reported an amount of revenue, it is the actual amount they earned: a kind of fiscal and mediatic honesty is presumed. But on the other hand, just because someone leaves a country to study does not mean that they were born there – they could have moved in their childhood – even if that would be true for most people.

The intellectual dance around the word "reasonable" is in itself delicate. It also implies a cultural dimension that is often forgotten or ignored: some cultures may not assume the same things we do. Regardless, it is rare that the interpretation matters a lot in the end-user scenario. For example, it is not absurd to suppose that the users of a question answering system will be culturally close to its designers. Even if interpretation errors remain, the system may be robust enough to still find the correct answer by combining several sources and methods. As a general trend, current evaluations arguably either focus on great precision of open inference, or good discrimination between a restricted set of possibilities of variable likelihood.

Beyond its simple definition, textual inference is a need that pervades almost every task in NLP. A relatively low level task such as syntactic parsing presents countless examples of ambiguous sentences which can only be resolved by having access to a context and drawing an inference from it.

(8) I saw the man with the binoculars.

(9) They are hunting dogs.

(10) He saw that gas can explode.

(11) We saw her duck.

The previous sentences are well-known examples exhibiting ambiguity. They are presented without context, which makes two semantic interpretations and parse trees possible, with no apparent most probable solution. In this next excerpt, from the movie *Animal Crackers* (1930):

(12) "One morning I shot an elephant in our pajamas. How he got in our pajamas, I don't know."

The first sentence is ambiguous but intended to be read as "while I was wearing our pyjamas, I shot an elephant". But the second one provides context and prompts the audience to backtrack and modify its interpretation. Immediately, context appears to be a very crucial notion for inference. This can then come off as a paradox: the flagship task for evaluating textual inference, RTE, tests systems mostly on simple pairs of short sentences – less than 10% of text or hypothesis have multiple sentences –, which do not make for a very rich context. But a context is not necessarily long in nature: in fact, knowing to pick the minimum context span sufficient to solve an inference correctly is a capability much desired of high level systems.

Coreference resolution has also been framed as an inference problem, with the Winograd Schema challenge (Levesque et al., 2012). A Winograd schema is a pair of sentences that differ in only one or two words and that contain an ambiguity –often in the form of a pronoun– that is resolved in opposite ways in the two sentences and requires the use of world knowledge and reasoning for its resolution. The schema takes its name from a well-known example by Terry Winograd (1972)

(13) The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.

If the word is “feared”, then “they” presumably refers to the city council; if it is “advocated” then “they” presumably refers to the demonstrators. This task is interesting because it is to our knowledge the only one which allows contexts to be swapped easily. It is all the more useful that the difference between the two contexts is often reduced to a single word.

In this section we wrote about the definition of the textual inference in NLP, some of its specifics, and we showed how it is required even for tasks which may be considered "low-level" or at least preliminary to solving any end-user problem. After a brief comparison with logical inference, we will describe more in detail the tasks whose purpose is to directly evaluate the inference capabilities of systems, like RTE or machine comprehension.

1.1.3 Link with logical inference

The word “inference” classically refers first and foremost to drawing *logical* conclusions, a central focus of the field of logic. There have naturally been attempts in the past to build on well-known logical theories, like first-order logic (FOL), to help in textual inference. However there are two difficulties inherent to bridging textual and logical inference. One is that it is difficult to accurately translate natural language into a coherent logical formalism (MacCartney and Manning, 2007). Some tricky linguistics features are not easily included: idioms, intentionality and propositional attitudes, modalities, temporal and causal relations, certain quantifiers, and so on.

The other is that it is unclear if textual inference – being much closer to being the reference for human inference – actually shares a lot with logical inference. It is commonly accepted that a system as strict and rigorous as first-order logic cannot be a satisfactory model for human reasoning (Johnson-Laird et al., 2015). Moreover, end-user scenarios involving inference are very much interested in conclusions that have a part of uncertainty. If a question-answering system cannot find an answer to a question, giving a set of likely answers is still much appreciated. The word “reasonably” in the traditional definition of textual inference thus rightfully allows an element of doubt in the conclusions drawn. Without it, systems get stuck drawing trivial facts from the premises and by-passing interesting but only highly plausible conclusions. This is why most models in logic and cognitive psychology for human inference nowadays build on probabilistic reasoning (Chater et al., 2006; Vityaev et al., 2013). Models for inference based on probabilistic logic have only very recently surfaced in NLP; they will be addressed later in the literature review.

1.2 Main inference-related NLP tasks

Even if the need for inference can be encountered in nearly all NLP works, only a few tasks actually evaluate directly a system's ability to reason on the text. "Directly" means that the need for some kind of reasoning is so immediate that even if we assume perfect and ideal subsystems taking care of classic processing like tagging, syntactic parsing, word-sense disambiguation or resolving coreference – all difficult tasks in their own right –, the problem is still not trivial, and is in fact, far from being solved. In this section, we present common modern tasks that assess a system's reasoning capabilities on text at various levels. The association of these tasks to the topic of this thesis is more or less acknowledged and studied; as it progresses, this section will unfold tasks of gradually less involvement of inference.

1.2.1 Recognizing Textual Entailment

Traditionally, RTE is the main task dealing with textual inference. The problem's definition, as it appears in Dagan et al.'s work in 2005 (Dagan et al., 2006), is simply stated:

> "We say that a text T entails a hypothesis H if, typically, a human reading T would infer that H is most likely true."

Other variants of this definition can be encountered in the literature, but they all come down to the same essential semantics that we discussed earlier. RTE finds it origin in other tasks like Question Answering (QA), Information Extraction (IE), (multi-document) summarization, and machine translation, which need a model for the phenomenon of variability of semantic expression: the same meaning can be expressed by, or inferred from, different texts. This phenomenon is the source for the bridging inferences of section 1.1.1.2. Even though different applications need similar models for semantic variability, the problem is often addressed in an application-oriented manner and methods are evaluated by their impact on final application performance, in an extrinsic way. This makes the systems difficult to compare and hard to re-use for other applications that might benefit from the same inference capabilities. Lastly, it splits the community so that researchers within one application area might not be aware of relevant methods that were developed in the context of another area of research.

The Recognising Textual Entailment (RTE) Challenge is then an attempt to promote an abstract generic task that captures major semantic inference needs

across applications (Dagan et al., 2006). It has been proposed every year in the period of 2005 to 2011, first by the PASCAL organization, then TAC, and more sparingly in recent times, with only the Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge at Semeval 2013.

The format does not change much between iterations: the participants are given training or development data, and then asked to submit a run of their system on previously unseen testing data. A piece of data is a simple pair of two short texts – most frequently single sentences –, the *Text* and the *Hypothesis*, and the system must tell if that pair is an entailment, that is to say that the hypothesis can be inferred from the text, or not. Some more fine-grained evaluations add a distinction inside the class of non-entailments: contradictions, where the hypothesis actually contradicts the text, and non-relations, where the two sentences are not related. The sentence pairs are extracted from various sources semi-automatically – most often from press articles – and annotated manually. Each year adds in general less than 1,000 pairs for training and the same amount for testing to the data pool.

The NLP community has been mainly interested in the simple binary problem, especially considering it as an information retrieval (IR) task: the most interesting pairs are those of valid entailment, and we want them selected and separated from the others. This translates into the choice of evaluation metrics for the systems. Popular evaluation methods include:

- accuracy: the simple proportion of correct calls (correct Yes' and No's)
- precision: the proportion of actually correct entailments out of all of those judged as entailments by the system
- recall: the proportion of correct entailments that the system indeed retrieved

We will describe evaluation methods more in details in the literature review.

Below are several examples of valid entailment pairs from the RTE 3 dataset:

(14) **T:** *In the Super Nintendo Entertainment System release of the game as Final Fantasy III , Biggs' name was Vicks.*
H: *Final Fantasy III is produced by the Super Nintendo Entertainment System.*

(15) **T:** *The black plague lasted four years and killed about one-third of the population of Europe, or approximately 20 million people.*
H: *Black plague swept Europe.*

(16) **T:** *Libya had been the target of US air raids in 1986, when the US had linked Libya to the bombing of a Berlin disco frequented by US servicemen.*
H: *The US has raided Libya.*

And some negative examples:

(17) **T:** *It is in the best interests of the Israeli government, the Israeli econoour, and Israeli citizens, for the government to negotiate a social security treaty with the US as soon as possible.*
H: *The US has made a treaty with Israeli government.*

(18) **T:** *David Beckham has announced he no longer wishes to be the captain of England's football team.*
H: *David Beckham plays for Real Madrid.*

(19) **T:** *Russia later imposed a hefty duty on oil exports to Belarus, claiming its neighbour was costing it up to 4bn in lost revenues each year.*
H: *Russia decreased the cost of oil exports to Belarus.*

1.2.2 Question Answering

QA is the classic NLP task of automatically providing answers to natural language questions from humans. As a natural and intuitive way for humans to interact with computers, QA's stakes for end-user applications are high. However, contrary to RTE, answering questions does not systematically require systems to be capable of inference. Whether they do depends on several factors: the type of the question, its structure and entities involved, the amount and nature of available data where the answer can be looked for. For example, consider the question:

(20) Why is water essential to life?

This is called a causal question, and is deemed to be one of the harder question types. But it becomes trivial if we find in our text corpus a sentence like:

(21) Water is essential to life because it helps carry proteins, carbohydrates, fats, vitamins, and minerals to each cell.

But the answer can be more complicated or even impossible to provide, if our system encounters only this bit of text:

(22) The reactions that form life as we know it need water, but there certainly could be life as we don't know it, that doesn't need water, but we don't know how to look for that.

In essence, the type of the question alone does not condition the need for inference. It is only in combination with the text available, which we can define as the context of the question, that the inference process becomes an interesting component of the problem.

With that being said, most QA evaluations in the past were concerned with factoid questions whose answer could be found with few other difficulties than the sheer mass of text available to the system. The QA track of the TREC evaluations (Voorhees, 2001), running from 1999 to 2004, is the very example of QA as information retrieval rather than inference task. QA which requires inference must be found in other much more recent types of evaluations, under the form of reading comprehension tests: a short text paired with a series of multiple-choice questions. Ironically enough, this evaluation format has been used throughout human academia – especially in anglo-saxon countries – for as long as standardized testing has existed. The reading component of SAT, taken by high-school students for college entrance in the United States, is a perfect example. For all their imperfections, multiple-choice questions provide a restricted framework to grade efficiently and rigorously, yet allow complex reasoning to be harnessed by students when answering the test.

QA4MRE, Question Answering for Machine Reading Evaluation, at the conference CLEF, has been the leading evaluation of this type since 2011. The QA4MRE task focuses on the reading of single documents and the identification of the answers to a set of questions. Questions are in the form of multiple choices, each having four to five options, and exactly one correct answer. The detection of correct answers might require the consideration of previously acquired background knowledge from reference document collections, but most importantly, various kinds of inference (Peñas et al., 2011). From 2011 to 2013, these questions and the associated answer candidates were extracted from the document semi-automatically, which made the task essentially evaluations for machines by machines, introducing biases in how the answer candidates and the question are formulated. The following example is from the reading tests from QA4MRE 2013:

Text:
[. . .]
Whereas EDM has achieved widespread popularity in Europe and the UK, it remains a comparatively underground phenomenon in the United States. The production and distribution of EDM has not been particularly adaptable to the American music industry, whose practices have been largely shaped by rock and pop. In addition, because EDM does not follow a standard song structure and often features very lengthy tracks, it does not easily lend itself to commercial radio.
[. . .]
Question: *Name a reason why EDM is not suited to the American music industry.*

1. EDM is influenced by rock and pop
2. **EDM often features very lengthy tracks (correct answer)**
3. EDM has achieved widespread popularity
4. EDM is produced from the beats up
5. none of the above

The text in the example is only a fragment of the original document, the fragment where the answer can be found. As we can see, most of the answer candidates are present in their exact surface forms in the text. The task is still quite difficult, because the right answer has to be linked to the expression of the question in the text, but this reading test does look machine-generated, which makes it an unsatisfactory way to evaluate systems aspiring to human reading capabilities.

Starting CLEF 2014, QA4MRE was reconducted but in name, to become the Entrance Exams task of the QA track (Peñas et al., 2013b). Piloted in 2013, as its name indicates, this task now evaluates computer systems taking entrance exams, specifically reading comprehension tests for English as a foreign language. The originality is that the tests are not machine-generated anymore, but extracted from real entrance exam subjects for future students at the University of Tokyo. The evaluation has become for machines by humans, much closer to the intended goal of machine reading comprehension at a human level. Consider the following example, an excerpt from Entrance exams 2015:

Text:
our husband hasn't stopped laughing about a funny thing that happened to me. It's funny now but it wasn't at the time. Last Friday, after doing all the family shopping in town, I wanted a rest before catching the train, so I bought a newspaper and some chocolate and went into the station coffee shop - that cheap, self-service place with long tables to sit at. I put our heavy bag down on the floor, put the newspaper and chocolate on the table to keep a place, and went to get a cup of coffee.
[...]
Question: *The woman telling the story*

1. always went shopping with her family on Fridays
2. **had been very busy and needed some time to recover (correct answer)**
3. wanted a newspaper and some chocolate to take home to her family
4. bought a newspaper and some chocolate so that she could keep a place at the table

Again, the text in the example is but the beginning of the actual document. As we notice, questions are now much more free-flowing and asked in a less formal way. However, as would be natural for testing students learning a foreign language, words of the text are seldom used in the right answer, and can be used as a misdirection device targeting the unskilled reader. The question can be useful, but as a trend, much less emphasis is put on the question than on distinguishing right from wrong answer candidates. This makes this task closer to RTE than before, interestingly, and the test overall does not have that machine-generated feel to it anymore.

We participated to this evaluation for the three years it has run. One problem that all systems encountered is the low amount of questions: between 50 and 60 each year. This made using machine learning, today's go-to choice throughout NLP, possible only for the third year, and the design of the system still had to take the low amount of data into account. An interesting line of research would be to build training corpora, but few additional reading tests of the same type are openly available.

1.2.3 Semantic equivalence

Several tasks tests systems on their ability to detect if two sentences are semantically equivalent. Paraphrase detection does precisely that and could very well be seen as a two-way RTE task: does sentence 1 entails sentence 2 and, at the same time, sentence 2 entails sentence 1? As much as this is a correct theoretical framing of the problem, the fact remains that paraphrase detection tasks offer less in the way of inference phenomenons than in simple syntactic rewritings and occasional substitutions of words with their synonyms. The following examples are two positive and two negative paraphrase pairs from the MSR Paraphrase Corpus (Brockett and Dolan, 2005), one of the de facto standard datasets for evaluating paraphrase detection, and were selected to showcase the most elaborate variability observed:

(23) **Sentence 1:** *The increase reflects lower credit losses and favorable interest rates.*
Sentence 2: *The gain came as a result of fewer credit losses and lower interest rates.*

(24) **Sentence 1:** *Dave Tomlin, assistant general counsel of The A.P., said his organization was still deciding whether to appeal.*
Sentence 2: *Dave Tomlin, AP's assistant general counsel, said the parties are deciding whether to appeal the order.*

(25) **Sentence 1:** *The Toronto Stock Exchange opened on time and slightly lower.*
Sentence 2: *The Toronto Stock Exchange said it will be business as usual on Friday morning.*

(26) **Sentence 1:** *The others were given copies of "Dr. Atkins' New Diet Revolution" and told to follow it.*
Sentence 2: *The researchers gave copies of "Dr. Atkins' New Diet Revolution" to the carb-cutters.*

Sentences in these pairs are unquestionably close in structure as well as lexicon. While they are good for evaluating basic handling of textual transformation processes in systems, they fall short of revealing the system's true capabilities for textual inference.

The more recent task of Semantic Textual Similarity (STS), piloted at Semeval 2012 (Agirre et al., 2012), asks the systems to grade on a continuous scale from

0 to 5 the semantic similarity of two sentences. Complete meaning equivalence is not required, and the annotation guidelines allow for some relaxation. The pairs which are annotated as not being paraphrases range from completely unrelated semantically, to partially overlapping, to those that were almost-but-not-quite semantically equivalent. In addition to demanding a more fine-grained assessment of sentence pairs, the data in STS corpora itself is slightly more complex in variability than usual paraphrase pairs, though this is done a lot by making the two sentences more or less irrelevant with each other, or by adding and subtracting details. In the following examples, from STS 2012, the first pair is rated 4.4, indicating almost complete equivalence, the second is rated 3.2, indicating rough equivalence, with important details missing, and the third is rated 0.8, not equivalent but possibly on the same topic:

(27) **Sentence 1:** *The problem likely will mean corrective changes before the shuttle fleet starts flying again.*
Sentence 2: *He said the problem needs to be corrected before the space shuttle fleet is cleared to fly again.*

(28) **Sentence 1:** *The American Express Corp. has pledged at least $3 million of more than $5 million needed.*
Sentence 2: *The city had requested federal funds, but withdrew that request when American Express pledged at least $3 million.*

(29) **Sentence 1:** *The technology-laced Nasdaq Composite Index .IXIC inched down 1 point, or 0.11 percent, to 1,650.*
Sentence 2: *The broad Standard & Poor's 500 Index .SPX inched up 3 points, or 0.32 percent, to 970.*

This task's methodology is rich and interesting, and STS seems poised to replace simple binary paraphrase detection as the semantic equivalence main evaluation. But the directional and structural aspect of textual inference, necessary to machine reading and the building of a mental model of the text, is quite absent.

1.3 A new hierarchy of inference classes

In the previous section, we presented several different NLP tasks which evaluate in a direct way the inference capabilities exhibited by computer systems. There it was already apparent that if the format is variable, the content itself has to be as

well. The broad definitions given both for cognitive psychology and NLP in section 1.1 do not say much of what information has to be drawn from the text, how to do it and whether it is an easy task or not. However, the ability to pinpoint the kind of inferential phenomena computer systems are realistically able to handle seems really valuable. One of our first contributions in this thesis is to establish a typology of inference classes as few have done before. Presented at LREC 2014 (Gleize and Grau, 2014), this work does not focus on the cognitive side of inference, at least not directly, nor on the content of inferences – which is what separating spatial and temporal inference represents, for example. We designed a hierarchical view of classes of inference with several goals in mind: the most important is being able to categorize instances of problems to spot where the difficulties lie and what nature and extent of NLP techniques and resources we need to leverage to handle them. It also allows to conduct more fine-grained diagnostic evaluations of systems, to complement typical black-box evaluations. In this section, we first present previous mentions of categorizing inference phenomenons in NLP contributions, we then expose a new taxonomy of inferences, which is finally supported by results of experiments on QA4MRE 2013's multiple-choice questions.

1.3.1 Literature on inference types and difficulty

To our knowledge, few attempts have been made to categorize the difficulty of inferences in a manner helpful to computational systems. The field of cognitive psychology (McKoon and Ratcliff, 1992) distinguishes bridging and elaborative inferences. Bridging inferences are drawn to fill the gaps in text and explicit –often through access to world knowledge– what is untold, yet required to understand what the text means. This is akin to what the machine reading system of Peñas and Hovy (2010) does: exploring domain-specific predicates and attempting to fill in missing arguments. Elaborative inferences are not required for textual coherence, and their status is strange in the context of NLP. As mentioned previously, it only takes a simple question to turn a potential elaborative inference into a much needed bridging inference, in the context of, say, answering reading comprehension tests.

Such a distinction, although insightful to anticipate the future design of machine reading and question answering systems, is still far from giving us hints on the automation of inference. Clark et al. (2012) provide a combination of well-known language resources used toward the common goal of textual entailment. Ablation tests are performed to compute the impact of each on the overall accu-

racy, but there is no way of knowing which resource helps best on a given type of questions or entailment pairs. Interestingly, Huang et al. (2013) take the angle of modeling human negative entailment capabilities. Finding in the text hints of a contradiction with the hypothesis appears to be an effective way to tackle textual entailment, but the tools and resources are still lacking to yield significant improvements over state of the art RTE systems. MacCartney and Manning (2007) introduce natural logic applied to textual inference. They describe the kind of inference problems this logic applied to natural language is capable of solving: natural logic handles monotonicity, in which the concepts or constraints expressed are expanded or contracted, but it is not designed to deal with paraphrase, temporal reasoning, or relation extraction. Posterior to our contribution, Weston et al. (2015) identify a set of 20 elementary subtasks in the problem of complex question-answering and present them in a similar way we ourselves do in the next section, by providing a simple example which cannot be solved without dealing with the related phenomenon.

More generally, contributions which analyze the categories of error of systems are those we want to build upon, altough quite rare as of late. (Moldovan et al., 2003) measure their system's accuracy by question class: factual, simple-reasoning, fusion-list, interactive-context, speculative. However, these classes are more about question types than they are about their difficulty. Even if the two are likely correlated in practice, question type is not theoretically tied to difficulty of the question (see section 1.2.2), which makes it a brittle indicator to rely on when attempting to gauge inference difficulty.

1.3.2 The hierarchy

Each of the classes is built to encompass a range of natural language semantic problems, tools, techniques, resources, and even human cognitive processes and levels of world knowledge required. As a definition, a problem is of a given class if it can reliably be solved within that class, but not within the classes below it in the hierarchy. The aim is to capture the inference phenomena that are sufficient and necessary to solve the problem. In this respect, those classes can be used in the same way complexity classes are used: both to characterize problems and the systems solving them –a system is of a given class if it can solve problems of that class but not problems from higher classes.

While this all sounds ambitious, this hierarchy does not pretend to be absolute, and in fact it does not need to: simply setting up the groundwork toward a unified framework to discuss the contrasts in relative nature and difficulty inside the same

problem –and sometimes the same dataset– is already helpful.

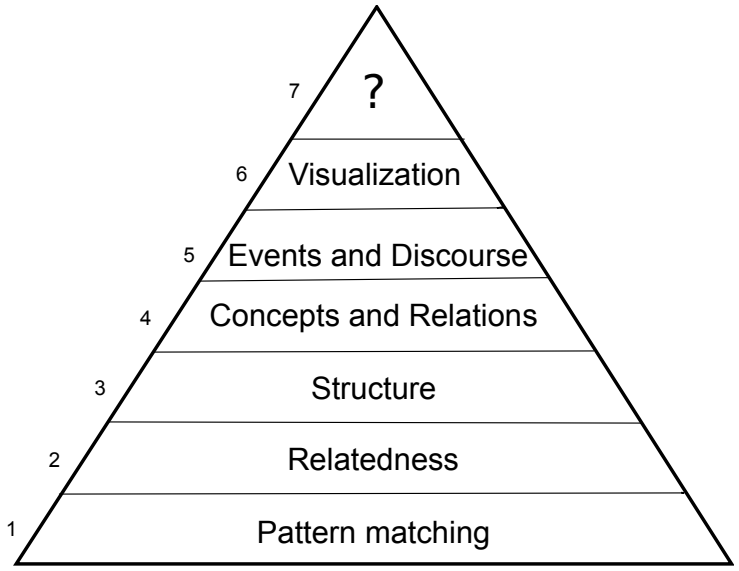

Figure 1.1: Hierarchy of inference classes

The hierarchy is shown in Figure 1.1. There are seven classes, and the first six (from bottom to top) pair up in three tiers, corresponding roughly to the units of sense considered. As members of the hierarchy, higher classes will often still rely on the techniques and resources featured in classes below, and it is reasonable to assume that a system of a given class behaves well when faced with problems of lower classes. The difficulty of the problems –and the error rate of the current state of the art methods– is expected to generally increase with the class.

Knowing what is difficult for machines seems key to us and is one of the goals addressed in this section. For each class, examples of representative problems are provided. We choose to frame the problems, when possible, as picking the most relevant item among several answer candidates to a natural language query. While it seems reasonable to expect that the classes can still characterize the absolute difficulty level of the query alone, this level would be difficult to assess without some elements of context or examples of concurrent wrong items the systems are susceptible to pick over the right ones. Providing this context and an explicit set of correct and incorrect items is a satisfying compromise which allows to showcase which techniques and resources are needed to discriminate reliably between possible answers to the query. In the examples, the right answer is marked by "correct".

It is then natural to specify how several well-known natural language tasks reduce to this common framework.

In *question answering*, the query can take the form of a natural language question, and items can be some choices of answer, the task being naturally to find a fitting answer to the question. Ranking answer candidates is also how state of the art question answering systems like IBM's Watson computer operate (Ferrucci et al., 2010), so this is a quite direct formulation of the task.

In *coreference resolution*, the query is a sentence containing a pronoun, associated with the text preceding it. The items are entities of the text and the task is to find which entity the pronoun refers to. Recently, (Levesque et al., 2011) have argued that the problem of resolving the difficult pronouns in a carefully chosen set of sentences, which he refers to as the Winograd Schema Challenge, could serve as a conceptually and practically appealing alternative to the well-known Turing Test (Turing, 1950). According to Levesque, the pitfalls lie in the difficulty of providing problems whose resolution is obvious for humans but hard for machines. We will use examples of Winograd Schemas to demonstrate that coreference resolution can reach a very high difficulty level. As a reminder, a Winograd Schema is a small reading comprehension test involving the question of which of the two candidate antecedents for the definite pronoun in a given sentence is its correct antecedent. More precisely, there is a word (called the *special* word) which appears in the question and sometimes the answer. When it is replaced by another word (called the *alternate* word), the text still makes sense, but the answer to the question changes. In this special coreference task, the query is set to the sentence and the question, and the candidate items are pairs of (special/alternate word, coreferent assignment), as shown for the classic Winograd scheme example in table 1.1. In the example, the test is to answer two questions, one with the word and the other with the alternate word, and link the 2 candidate entities so that they answer the right question.

Text: The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.
Question: Who [feared/advocated] violence?
1) feared $\rightarrow$ councilmen, advocated $\rightarrow$ demonstrators
2) feared $\rightarrow$ demonstrators, advocated $\rightarrow$ councilmen

Table 1.1: Winograd schema sample

In *student answer analysis*, an applicative task to Recognizing Textual Entail-

ment evaluated in SemEval 2013's Task 7 (Dzikovska et al., 2013), the query is a correct reference answer to a question, and items are correct and incorrect student answers. The task is to find correct student answers matching one reference answer. Some of the related examples are sampled from Semeval 2013 Task 7's Beetle test data.

Other problems which may fit this framework are word sense disambiguation and entity linking.

In the next sections, we present the three Tiers and their classes, with definition and examples. The first tier deals with phenomena at the word level, the second tier deals with the sentence level, and the third tier goes beyond the text.

1.3.2.1 Tier 1: Word level

Class 1, *Pattern matching*, pertains to problem instances that can be solved using only words of the text, without any intent to capture the representation of a higher-level structure like the sentence, and using minimal world knowledge. Sentence chunking to create smaller groups of words, filtering words of the text (with a stop-word list, for example), tokenization, grouping words in n-grams, basic stemming and lemmatization are the most advanced text processing needed to solve this class of problems.

Text: Bob is going to the swimming pool. Question: What is Bob going to do? 1) Eat a sandwich. 2) Go swimming. (correct)

Table 1.2: Question Answering (Class 1)

Reference answer: Terminal 4 and the positive terminal are separated by the gap 1) Because they aren't damaged. 2) positive battery terminal is separated by a gap from terminal 4 (correct)

Table 1.3: Semeval 2013 Task 7 (Class 1)

These sample problems (tables 1.2 and 1.3) are easily solved through counting the number of words common to a choice and the given text. It is enough to discriminate between the right choice (choice 2) and something irrelevant.

Class 2, **Relatedness**, does not need the inference process to represent a higher level of sense than Class 1 (we stay at word level), and uses mostly the same processing tools, but we add the notion of lexico-semantic variations of words to capture a shallow notion of semantic relatedness. Lists of synonyms/hypernyms, taxonomies, thesauri –including WordNet (Miller, 1995) used in the simplest ways– and dictionaries are among common resources that are added in to deal with those problems.

Text: Bob bought this hamburger. Question: What has Bob done? 1) Obtained a sandwich. (correct) 2) Played in the water.

Table 1.4: Question Answering (Class 2)

In the example (table 1.4), counting the common words between the choices and the text is not enough and does not distinguish one answer from the other. But *hamburger* is an hyponym of *sandwich*. This can be learned from a good enough synonym/hypernym resource, like WordNet, or the Wiktionary [1].

1.3.2.2 Tier 2: Sentence level

Inference problems of Class 3, **Structure**, require the capture of some notion of higher-level structure in the text, at least higher than the word, generally at sentence –or clause– level. Several of the most well-studied text processing techniques are useful, like part-of-speech tagging, chunking, syntactic parsing, predicate-argument extraction, semantic role labeling, or polarity detection. Note that techniques and resources mentioned in Tier 1 are often still needed, but there is now the extra need to know not only what the words are, but what roles they occupy relatively to the others in the sentence.

Text: As it's raining today, Bob won't go to the beach. Question: What is going to do Bob? 1) Stay home. (correct) 2) Go to the beach with his friends.

Table 1.5: Question Answering (Class 3)

[1] `http://www.wiktionary.org/`

Reference answer: Terminal 4 and the positive terminal are separated by the gap 1) terminal 4 is not connected to the positive battery terminal (correct) 2) because there is no gap between terminal four and the positive terminal

Table 1.6: Semeval 2013 Task 7 (Class 3)

The problem described in the examples Tables 1.5 and 1.6 is as expected harder than Tier 1 problems. Simple Tier 1 techniques can work against the system and reliably pick the wrong choice. In table 1.5, we have to determine that no matter what, Bob will surely not *go to the beach* , by parsing the text and detecting that this clause is negated, so we pick the other answer. This is also an example of using negative entailment in the decision process. In the example table 1.6, to pick the student answer corresponding to the reference answer (choice 1), polarity detection is used on some kind of predicate-argument structure extraction to establish typed links between the *terminal 4* and *the positive terminal*. Note that it is necessary to know that *separated* and *connected* are antonyms, which can be found in resources used in Class 2.

Similarly to the first two classes, Class 4, **Concepts and relations**, takes the complexity of Class 3 and adds in extended world knowledge. Establishing a link between a word –or sequence of words– and a real-world concept is now required, as well as using not only the relations between concepts discovered in the text, but also those stored in a background knowledge base. Named entity recognition, paraphrases, ontologies, relations and properties extracted from dictionaries, WordNet (Miller, 1995) or Wikipedia and even relations extracted through web-crawling like TextRunner (Yates et al., 2007) are good techniques and resources to capture concepts and relations. Some basic reasoning engine may be needed.

Text: Bob once gave his sister Alice a pendant. Question: Who gave Alice the pendant? 1) A brother. (correct) 2) A sister.

Table 1.7: Question Answering (Class 4)

In the question answering example (table 1.7), a system has to know that if

Text: The lawyer asked the witness a question, but he was reluctant to [answer/repeat] it. Question: Who was reluctant to [answer/repeat] the question? 1) answer → witness, repeat → lawyer (correct) 2) answer → lawyer, repeat → witness

Table 1.8: Winograd schema (Class 4)

a male individual has a sister, then he is the brother of that female individual, which is not linguistic knowledge easily obtainable in the previous classes. In the Winograd scheme (table 1.8), there are two main ways to go about it. Either the system knows that the indirect object of the verb *ask* often has to answer the question after it is asked, which is quite complex reasoning that will be handled in class 5, or it just picks the character that is more likely to answer questions, when a *witness* and a *lawyer* are involved. For this method and this example, TextRunner gives as potential relations between *witness* and *question* only variants of *answer* or *ask* at the passive form, while relations found between *lawyer* and *question* are much more diverse, modeling the fact that the witness is often one who strictly answers questions.

1.3.2.3 Tier 3: Beyond the text

The phenomena of Class 5, **Events and discourse**, go beyond a single sentence and deal with characters and events in the text. In particular, understanding the overall structure of a sequence of several sentences is required: to know when an event might have happened without being mentioned in the text, what event might happen in the future, what roles are filling the characters. NLP techniques and resources at this stage are much more scarce. Discourse parsing can unveil simple causal or temporal relations between events. Event chains can help filling out the blanks in a succession of events. It is likely that common-sense knowledge of human society and interactions would help at this stage, without needing the full extent of what Class 6 is about.

All of the choices in the example (table 1.9) are true statements, but are not the reason asked in the question. This question also deals with common states of mind of human beings.

The Winograd schema of class 5 (table 1.10) requires keeping track of two characters and their interaction in the text. Although technically all the informa-

Example from QA4MRE 2013, entrance exams task
Text: I probably would have [continued to argue with her], but as I lay there, I could tell that Susan's phone call was not good news. I knew she had a boyfriend back home. From what I could hear her say, I guessed he had found a new girlfriend. Question: Why was Susan so upset by the phone call? 1) Mary's boyfriend had found a new girlfriend. 2) The call interrupted her argument with Mary. 3) Her boyfriend had lost interest in her. (correct) 4) The caller did not want to talk to Mary.

Table 1.9: Question Answering (Class 5)

Text: Susan knew that Ann's son had been in a car accident, [so/because] she told her about it. Who told the other about the accident? 1) so $\rightarrow$ Ann, because $\rightarrow$ Susan 2) so $\rightarrow$ Susan, because $\rightarrow$ Ann (correct)

Table 1.10: Winograd schema (Class 5)

tion is included in the same sentence, there are multiple clauses to link together. Detecting a causal relationship –with reliable directionality– is a typical class 5 problem.

Class 6, **Visualization**, goes beyond the text itself, and requires an actual model of the situation at hand (Zwaan and Radvansky, 1998). A few dedicated NLP applications exist solely to solve one of the many facets of Class 6 inferences, including: temporal and spatial reasoning, sentiment detection. In general, computer systems are lacking human senses –vision and hearing– to deal with these problems in a generic fashion.

The example from table 1.11 is one of the harder questions. One has to know or evaluate the predicted human population of Earth by 2050 and make a computation about that number. Several choices are provided so we can pick the likelier number, but the system still has to be aware of global growing demographics on Earth, and know how to perform simple calculations. Alternatively, one has to guess that the other likely number (85 million) is probably just a decoy because the number 85 is present in the text but not to designate a population count; this

Example from QA4MRE 2013, main task
Text: 1 person in 85 will be affected [by Alzheimer's] by the year 2050. Question: How many people affected by Alzheimer's are there expected to be in the year 2050? 1) 123 million. (correct) 2) 85 million. 3) none. 4) a pretty small number. 5) None of the above.

Table 1.11: Question Answering (Class 6)

Text: I tried to paint a picture of an orchard, with lemons in the lemon trees, but they came out looking more like [light bulbs / telephone poles]. Question: What looked like [light bulbs / telephone poles]? 1) light bulbs $\rightarrow$ lemons, telephone poles $\rightarrow$ lemon trees (correct) 2) light bulbs $\rightarrow$ lemon trees, telephone poles $\rightarrow$ lemons

Table 1.12: Winograd schema (Class 6)

kind of meta-reasoning about the task format (multiple choice questions) is not currently handled by systems. It is not intended to be the main natural language problem to be addressed in the near future, being understandably of less interest in the grand scheme of things. In the Winograd schema from table 1.12, it is much easier for a human to just imagine what fruits or trees might look like –relatively to their visual shape– than for a computer.

1.3.2.4 Leaving room for the unexpected

We chose to leave room for more complex inference problems and techniques that may arise.

The previously defined classes are intended to help categorize systems and datasets. In the next section, we perform experiments which demonstrate the value of our classes with respect to this goal.

1.3.3 Experiments

In these experiments, we define an annotation task on reading comprehension tests. For each question, we annotate both the relevant passage and the answer choice in term of the lowest estimated class of systems it requires. Two different corpora are annotated, one made automatically to evaluate machines, and one made manually to evaluate humans. We show that the second corpus contains questions of higher classes – hence harder according to our taxonomy – than the first one.

We first present the multiple-choice question answering dataset. It consists in the question answering test sets at QA4MRE 2013 (Peñas et al., 2013a). There are two tasks, the Main task and the Entrance Exams task, and both feature the same format: a series of long texts, and for each of them, several multiple-choice questions to answer. The Main task's questions are traditionally designed to evaluate natural language processing systems. But the all new Entrance Exams task features tests of English as a foreign language at the Japanese University Entrance exams, hence this dataset is designed to evaluate humans. This is a key difference.

There are 284 questions over 4 topics of 4 reading documents each in the Main task, and 45 questions over 9 reading documents in the Entrance Exams task. Questions have 5 answer choices in the Main task (including a *None of the above* option to indicate that none of the provided answer choices is correct), 4 answer choices in the Exams task (a *None of the above* option is not present for those). Each question has been annotated with its correct answer (as provided by QA4MRE organizers) and our own annotations of a 3-sentence passage in the text containing a single answer sentence. We removed questions for which we couldn't find a correct passage: in particular, all questions where *None of the above* is the correct answer were filtered out (39% of the Main dataset).

Our goal is to annotate each question with their class in our hierarchy. We turn the question answering problem into 2 separate subtasks: first we need to find the passage containing the answer, and once this is done –essentially reducing the search space for the answer from hundreds of sentences to just three– we need to consider answer candidates present in this passage, and choose the correct one. Each of these tasks can be framed as described at the beginning of section 1.3.2. For the passage retrieval subtask (PR), the query is the question and the candidate items are the passages of the text. For the answer choice subtask (AC), the query is the question and the 3-sentence passage –now assumed correct and containing the answer– and the candidate items are the answer choices. The overall difficulty of the question answering task (QA) can be approximated by taking the maximum

of the difficulties of the two subtasks, passage retrieval and answer choice [2].

For convenience purposes, we run a baseline counting the common lemmatized non stop-words between candidate items and query and rank the candidate items according to their score. The tokenizer and lemmatizer used are part of the Stanford CoreNLP tagging tool (Toutanova et al., 2003). We then annotate for each question the class which corresponds to the passage retrieval part and the class which corresponds to the answer choice part, that is to say, the class that is necessary and sufficient to distinguish the correct item from the incorrect ones. When our baseline ranks the correct passage in first place, we automatically annotate this passage retrieval step as being of class 1. Similarly, when our baseline ranks the correct answer in first place –that is to say, the correct answer has strictly more words in common with the correct passage than all the other candidates–, we automatically annotate this answer choice step as being of class 1. The rest of the classes are manually annotated by two annotators on the Main task and the Exams task.

A Cohen's Kappa score of inter-annotator agreement is computed (Cohen, 1968). We obtain 0.81 on the passage retrieval subtask and 0.86 on the answer choice subtask for the Main dataset, on 51 questions annotated by both annotators, which denotes very high agreement. However, we only obtain a Cohen's Kappa of 0.49 on the passage retrieval subtask and 0.57 on the answer choice subtask for the Exams dataset, on 30 questions annotated by both annotators, which is a much lower agreement.

Table 1.13 reports the class distribution for all three tasks (PR, AC, QA) on both datasets, Main (M) and Exams (E). We observe that the class distribution for the overall question answering task is different for Exams questions, even though the problem and format are the same. Exams contain significantly more questions of Tier 3 compared to Main. And among Tier 1 classes, they exhibit five times as much Class 2 phenomena as Class 1, whereas Main's first two classes are balanced, with even a slight bias toward simple pattern matching questions. This confirms the hypothesis that the nature of questions in the Entrance Exams corpus is different from that of the questions in Main. Tests for entrance exams use more reformulations of the text (pertaining to Class 2 inferences) to test the english vocabulary of the student. The questions can also involve cognitive processes of

[2]In practice, we likely misestimate the difficulty of the passage retrieval subtask, because actual passage retrieval as performed by systems can benefit from including words of the right candidate answer, but likewise can be noisier as we add words of the wrong candidate answers. Nevertheless, a human reader can find most of the relevant passages without reading the corresponding provided candidate answers, so separating the task in those 2 subtasks is reasonable.

Class	PR (%)		AC (%)		QA (%)	
	M	E	M	E	M	E
1:Pattern matching	55	38	52	33	24	4
2:Relatedness	12	15	6	16	16	19
3:Structure	16	13	32	13	37	17
4:Concepts and Relations	4	0	0	2	2	2
5:Events and discourse	14	32	8	22	18	35
6:Visualization	0	4	2	13	2	17
7:?	0	0	0	0	0	0

Table 1.13: Class distribution for Main (M) and Exams (E), annotated by annotator 1

Annotators	Annotator 1		Annotator 2	
Annotators	PR	AC	PR	AC
1:Pattern matching	50	33	50	33
2:Relatedness	15	13	0	17
3:Structure	8	10	4	7
4:Concepts and Relations	0	3	19	27
5:Events and discourse	27	20	23	7
6:Visualization	0	20	4	10
7:?	0	0	0	0

Table 1.14: Class distributions from 2 annotators for Exams (documents 1 to 7)

higher order (Class 5 and more) which make heavy use of common-sense knowledge, as it is assumed to be naturally available to a human being.

The table 1.14 attempts to provide some insights on why our inter-agreement rate is much lower on the Exams task than on the Main task. We found that for more difficult problems, it can be confusing to pinpoint the class to which a phenomenon belongs. For example, we report that there seems to be confusions between Class 2 and Class 4 at the passage retrieval level. Class 2 introduces the notion of semantic relatedness and exterior knowledge can already at this level be captured through the resources employed, which makes it close to Class 4 in term of external world knowledge.

Nevertheless, the class distribution of Entrance Exams being skewed towards the upper Tier 2 and Tier 3, compared to Main's distribution, we could expect the performance of systems which ran on both at QA4RME 2013 to be lower because

the problem is overall harder. This is indeed the case, as seen in Table 1.15: we report the accuracy of three systems at the evaluation across the two tasks, with respect to the class distribution already mentioned. They lose about 30% accuracy when going from the Main task to the Entrance Exams task. The accuracy loss is effectively even bigger, because the Entrance exams only feature 4 choices a question, compared to 5 choices for the Main task.

Systems	Main (accuracy)	Exams (accuracy)
jucs	0.59	0.42
nara	0.33	0.22
limsi	0.28	0.22
Class	Main (%)	Exams (%)
1:Pattern matching	24	4
2:Relatedness	16	19
3:Structure	37	17
4:Concepts and Relations	2	2
5:Events and discourse	18	35
6:Visualization	2	17

Table 1.15: Accuracy of systems at QA4MRE 2013 and question classes, for both tasks

1.3.4 How to use this going forward

This section presents a new hierarchical taxonomy of textual inferences. This taxonomy is designed with computational systems in mind, and encompasses tasks, techniques, tools and resources. As seen in the experiments, it can indeed make the contrasts within a dataset apparent, both in term of nature and overall difficulty of the task. From there we can evaluate systems in a much finer way, and really get a grasp of the classes they are designed to handle. We can then guide further improvements on systems and choose the kind of problems we want to concentrate on.

1.4 Conclusion

This chapter presented our motivation for this thesis. We introduce textual inference from various angles, from human inference to logical inference. We de-

scribe practical NLP problems related to inference as well as some of the standard datasets on which systems are evaluated. The end of the chapter presents one of our contribution establishing classes of textual inferences according to their difficulty. Going into the literature review, the reader should now have some knowledge of the main problems the systems are faced with when handling textual inference in actual NLP tasks. The different classes of our taxonomy highlight what is generally currently done about them and what is definitely missing in the future. our literature review loosely follows the progression hinted by those classes, especially in how structure and background knowledge are added in increasingly complex systems.

Chapter 2

A literature review of automatic textual inference

Research on inference has officially existed in the NLP landscape since 2005, when RTE was introduced (Dagan et al., 2006). This is a bold statement, and of course, references to individual attempts at the resolution of complex inference phenomena can be found before that point. But overall, the collective effort toward automatic inference was only really kick-started once the research community had a common framework to design and evaluate dedicated systems.

This literature review aims at being agnostic with respect to the task. As we showed in the first chapter, many tasks can be used to evaluate textual inferences, and we intend to unify contributions on different tasks in term of the nature of the methods they use. The first part deals with evaluation, with a quick look at datasets, and the definition of popular evaluation measures. The second part presents the actual contributions, following the hierarchical structure introduced in 1.3.2.

2.1 Evaluation

2.1.1 Datasets

Intrinsic evaluation in tasks pertaining to textual inference always has the same format. The system has to automatically label pairs of short fragments of text with a relation. Oftentimes, these fragments are single sentences, and the possible relations quantify a degree of inferability existing between them. For example, in paraphrase identification, the two sentences are either semantically equivalent or they are not. In RTE, a common task is to assert if there is an entailment relation between two sentences, or if there is not. In those examples, where only two choices are available, the problem is called a binary classification. Although binary classification tasks certainly are the most frequent evaluations in this field, sentence pairs can be labeled with a more fine-grained set of relations. For example, at Semeval 2013, the RTE task (Dzikovska et al., 2013) presents pairs with a reference answer to a question, and a student answer, with the goal of assessing the correctness of the student answer. Systems have the possibility to classify the pairs with five different labels: correct, partially correct but incomplete, contradictory – the student answer contradicts content in the reference answer –, irrelevant – it does not contain information directly relevant to the answer–, or not in the domain. The "contradiction" label in particular is often the most interesting judgment to add to the basic set of binary alternatives, because it tests systems on their ability to accurately detect a breach of coherency between two sentences.

Evaluation campaigns build corpora of hundreds of pairs of sentences, possibly thousands. The one common variable in the constitution of a corpus is that pairs are annotated manually: the task is indeed too difficult to reliably automatize the creation of gold standard sentence pairs both positive and negative in term of inference. This is especially true when we want two sentences in an entailment relation, but not semantically equivalent. Nonetheless, the sentence pairs can be automatically generated or extracted from another corpus before being presented to a human annotator. This is the way datasets for the RTE campaigns have initially been gathered. For the latest installment at Semeval 2013 however, the reference answers were the answers given by teachers to a question about their subject, and the student answers were given by students to the same question during a real electronic test. This is what we could call fully manually built data, corresponding to real-life situations and needs, but this kind of efforts remains very costly, and hence, very rare.

Extrinsic evaluation of textual inference can use datasets of various other for-

mats, and it would be difficult to present them in exhaustive details. An example that we already mentioned is the Winograd Schema challenge (Levesque et al., 2012), a coreference resolution problem requiring inference. The one format that is interesting to this thesis and to the machine reading community in general is reading comprehension MCQs. Invariably, the system is faced with a set of texts, and for each of them a series of questions with a fixed number of answer choices –usually between 4 and 6. Again, semi-automatic approaches to corpus constitution existed at first, as seen in CLEF 2011 to 2013 with the QA4MRE challenge (Peñas et al., 2011), but recently, questions for real human test takers have been tackled, with the Entrance exams track starting from CLEF 2013 (Peñas et al., 2013b). There is obviously an incredible wealth of such tests, because the format has been used for a long time throughout the anglo-saxon education system. However, it is paradoxically hard to find a sizable amount of free and open-access test samples, because they represent an important commercial interest: past tests are actually sold to students each year for practice. At the time of writing, there is still no available large-scale annotated dataset of reading comprehension tests.

2.1.2 Classification measures

Evaluations tend to have first and foremost a quantitative nature. We usually like to have a single number that characterizes the performance of a system on a dataset. This allows immediate and straight-forward comparison with other systems. Several different measures can be used, depending on the task and the goal of the system. In classification tasks, either the system annotates a test item with the correct label, or it chooses another label among the others. The simplest measure is the *accuracy*, defined as the fraction of correct labels assigned by the system.

In binary classification, two other measures originating from information retrieval are popular: precision and recall. To define these, let us present four notions:

- *True positives* are elements that were predicted by the system as positive (meaning, for example, "this is a textual entailment"), and that are indeed positive in reality.
- *True negatives* are elements that were predicted by the system as negative ("this is not a textual entailment") and that are indeed negative in reality.

- *False positives* are elements that were wrongly predicted by the system as positive. They are negative in reality.
- *False negatives* are elements that were wrongly predicted by the system as negative.

Precision is then defined by Equation 2.1.

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} \tag{2.1}$$

Recall is defined by Equation 2.2.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{2.2}$$

Precision is interested in the system being able to say confidently when an element is positive. If the system ignores some positive elements and misclassifies them, precision is not penalized. In term of textual inference, a high-precision system can be used to draw the most trustworthy inferences and ignore everything less sure. On the other hand, recall is interested in the system being able to retrieve a lot of positive elements. If the system also captures negative elements by misclassifying them, recall is not penalized. In term of textual inference, a high-recall system will cover a lot of the complete set of correct inferences.

A combined measure, called *F-measure*, or F-score or F_1 score, is the harmonic mean of precision and recall, obtained by Equation 2.3.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.3}$$

Ideally, a system must have both high precision and high recall, but it is also common to purposely design systems to achieve good precision at the cost of recall – and vice versa –, as a compromise when obtaining two high values is not feasible. When the system has a parameter whose value has an impact on precision and recall, it is interesting to draw a *precision-recall curve*: plotting the 2D graph of precision on one axis, recall on the other, obtained for configurations of the system corresponding to different values of the parameter.

When working with multiclass classification, i.e. with more than two labels, it is possible to compute a pair of precision/recall values for each of the classes, and then micro-averages and macro-averages of these measures, but as the literature and this thesis both focus mainly on binary classification problems, we choose to not detail these here.

2.1.3 Ranking measures

Ranking measures are used in the context of ranked information retrieval: when the task asks for the production of a ranking of candidates for each of its items –also called queries–, or when the system's inner workings involve outputting a ranking, even when the task does not require one. While inference problems are without a doubt better framed as classification problems, common ranking tasks like question answering and document/passage retrieval can still require a great deal of inference making. As in the previous section, different measures evaluate solutions to different needs and goals.

The mean reciprocal rank (MRR) is the inverse of the rank of the first relevant ranked candidate, averaged over all the items in the testing data, as shown in Equation 2.4.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \tag{2.4}$$

with rank_i the rank of the first relevant document for query i.

The mean average precision (MAP) is defined in Equation 2.5.

$$\begin{aligned}\text{AveP} &= \frac{\sum_{k=1}^{n}(P(k)\times \text{rel}(k))}{\text{number of relevant documents}}\\ \text{MAP} &= \frac{\sum_{q=1}^{Q}\text{AveP(q)}}{Q}\end{aligned} \tag{2.5}$$

where $P(k)$ is the precision up to rank k and $\text{rel}(k)$ is an indicator function equaling 1 if the item at rank k is relevant, 0 otherwise.

MRR is a measure which focuses only on the ability of a system to retrieve at least one relevant candidate with a high rank, the relevant items below the first are ignored. MAP must be used when there are multiple relevant candidates that we are interested in retrieving.

Precision at n (p@n) is the fraction of items where the system retrieved one relevant candidate among the first n of the ranking. In particular, p@1 is the fraction of the test items for which the system gave the correct answer in the first position. In the context of a real-world task like web search, a low p@1 does not necessarily mean that the system cannot be used, if its p@3 or even p@5 are good enough, because the end user can afford to quickly assess 5 web page titles before picking the one to navigate to. In contrast, in the context of factoid question answering, p@1 is a very important measure for end users: any wrong answer before the correct one will confuse the end user and potentially lead him on with false information.

This ends our tour of the most commonly used evaluation measures. The next section describes the contributions related to inference found in the literature, organized by the class of linguistic features they rely on.

2.2 Lexical approaches

The rawest view of written text is that of a succession of characters representing graphemes. However, this view does not help in any way the resolution of high-level semantic problems. The first step toward making sense of the text is to extract information of a more helpful nature. This information is often called the *features* of the text in the context of machine-learned NLP, and we will keep using this denomination even when machine learning is not used in the described work. For each kind of feature, we present here previous work making use of it to solve inference-related problems.

Words form the most basic unit of sense in most natural languages. As such, the intuitive solution to the problem of establishing any semantic relation between two sentences is to just compare their words in some way. The methods of this section do not assume any other structure in the sentence than its pure sequentiality. And although they are quite low in our hierarchy of inference problems, these methods are currently still used very effectively to obtain state-of-the-art performance on a lot of tasks.

2.2.1 Lexical overlap on surface forms

Surface forms are words as they appear in the text. Methods based purely on them require minimal NLP machinery and lexical resources, but also typically are not effective for high-level tasks. In today's inference tasks, they are used as weak baseline systems, and represent what can be obtained without injecting additional knowledge about words in the process.

2.2.1.1 Counting in bag-of-words

There are several ways to extract surface form features from the text. At its simplest, a sentence can be viewed as a so-called *bag-of-words*, the set of its words. Simple features can be the presence or the number of occurrences of a given word in the sentence, but they are seldom used in textual inference-related fields, due to the limited size of datasets. More commonly, the similarity of two sentences can

be measured by the number of words they share, which is the size of the intersection of their bag-of-words. These "counting" features are often normalized, so as to not give an unfair advantage to long sentences. One can choose to normalize by the size of the union of both sentences – the Jaccard index (Rajaraman et al., 2012) – or the length of either sentence for a more directional measure. This kind of measure can seem overly simplistic at first glance, but remains relatively effective, with respect to how little effort it takes to implement. In recent work, Yih et al. (2013) use this exact measure as a baseline to solve a problem of *answer sentence selection*, essentially sentence ranking for QA, on past TREC data gathered by Wang et al. (2007). An example of question and answer candidates, with annotations of correctness, can be found below.

Question: *What do practitioners of Wicca worship?*
Candidate answer sentences:

1. An estimated 50,000 Americans practice Wicca, a form of polytheistic nature worship. [correct]
2. The inch-thick chaplain handbook includes a five-page primer on Wicca, described as "a re-construction of the Nature worship of tribal Europe." [correct]
3. Wicca – sometimes spelled Wycca – comes from the Old English word for witch. [incorrect]
4. ...
10. "When people think of Wicca, they think of either Satanism or silly mumbo jumbo." [incorrect]

Yih et al. (2013) report that it yields a MAP and MRR of 0.57 and 0.63. This is to be compared with the random baseline (answer sentences are ordered at random) which yields 0.40 and 0.49, and the actual system described in the article, which yields 0.69 and 0.79, the best at the time. On MRR, this means that the simple baseline based on word counts does 43% better than random, and the best system only improves that baseline itself by 25%, which is not as great of a leap. We will see that there are very straight-forward ways to improve on that baseline to bring it even closer to the state-of-the-art method.

2.2.1.2 A string of words

To go beyond IR-like tasks toward inference, it is necessary to account for the sequential nature of sentences. Similar counting features as above can be used, but on n-grams, the subsequences of n consecutive words of the sentence. The *Levenshtein distance* counts the number of elementary operations – insertion, deletion, substitutions – necessary to turn one sequence of words into another (Levenshtein, 1966). These features offer more diversity and combinatorial complexity than the previous unigram features, and as such, are used in more varied and effective methods. It is also notable that a lot of approaches based on these features draw inspiration in the field of machine translation, especially their evaluation measures like BLEU (Papineni et al., 2002). To solve paraphrase identification, Madnani et al. (2012) propose a classifier with machine translation metrics as sole features. Some of the basic metrics they use rely only on surface n-grams, like BLEU, NIST and TER rely only on surface n-grams. BLEU computes the amount of n-gram overlap –for different values of n – between one sentence and another reference sentence. NIST (Doddington, 2002) is a variant of BLEU that uses the arithmetic mean of n-gram overlaps, rather than the geometric mean. It also weights each n-gram according to its informativeness, as indicated by its frequency. TER (Snover et al., 2006) is a measure counting the number of *edits*, operations like insertion, deletion and substitution of a word meant to transform a sentence into another. It accounts for sequentiality with a heuristic algorithm to deal with shifts of n-grams. A classifier is learned on training data, with different set of those measures as features. This work is evaluated on MSRP, the Microsoft Research Paraphrase corpus (Dolan et al., 2004), one of the standard dataset to evaluate paraphrase identification. Two examples, the first one, positive – a true paraphrase – and the second, negative, can be found below. Other examples can be read in Section 1.2.3.

(30) **Sentence 1:** *The virus kills roughly 36,000 people in an average year, according to the U.S. Centers for Disease Control and Prevention.*
Sentence 2: *Complications from flu kill roughly 36,000 Americans in an average year, according to the U.S. Centers for Disease Control and Prevention.*

(31) **Sentence 1:** *In midafternoon trading, the Nasdaq composite index was up 8.34, or 0.5 percent, to 1,790.47.*
Sentence 2: *The Nasdaq Composite Index .IXIC dipped 8.59 points, or 0.48 percent, to 1,773.54.*

This dataset contains 66.5% of positive paraphrases, which sets the accuracy for an "all-paraphrase" baseline. Table 2.2 (p. 81) summarizes the results of all the relevant contributions on this dataset. The classifier using only the three basic evaluation measures presented obtains an accuracy of 74.1%, beating several systems from up to 2008. Their best result, including modern measures with lexico-syntactic resources, obtains 77.4%. Again, we see that surface form features can perform quite well on inference-related tasks.

Techniques based on string of words are still employed in recent years, like at SemEval 2013. SemEval 2013 was a special edition of the conference on semantic evaluation, at least as far as textual entailment is concerned. Its Task 7, entitled Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge (Dzikovska et al., 2013) consists in automatically evaluating student answers to a question, given reference answers. As its title indicates, this evaluation frames the task as a problem of RTE, and is in fact the latest edition of the RTE challenge to date. It marks a break with earlier RTE challenges, because the dataset is quite different. While RTE 1 to 7 present datasets of pairs of sentences, often extracted from news articles, sentences from Semeval 2013 Task 7 are answers from real high-school students to questions, with reference answers from teachers. Questions often pertain to very specific subjects like physics or electronics problems. The lexicon, also domain-specific as a consequence, is limited enough that using external resources does not prove useful, as demonstrated by the top systems at the evaluation. The task also evaluates on two, three or five judgments, or levels of entailment. Two or three (with the relation "contradictory") are quite common in RTE challenges, but five is a novelty. The student answer can be *correct*, *incomplete*, *contradictory*, *irrelevant* – when it does not address the question topic – and *not-in-the-domain* – when it goes outside the question-answer frame, like "I don't understand". An example of question, with annotated student and reference answers can be found below.

Question: *Explain why you got a voltage reading of 1.5 for terminal 1 and the positive terminal.*
Reference answers:

- Terminal 1 and the positive terminal are separated by the gap.
- Terminal 1 and the positive terminal are not connected.

Student answers:

- because they are connected to opposite terminals [correct]
- because there is a gap [incomplete]
- because it's connected to the positive terminal [contradictory]
- because 1.5 v minus 0 v equals 1.5 v [irrelevant]
- It was a mistake. I looked at it wrong. [not-in-the-domain]

The two best systems are interesting in that they do not use any external lexical, syntactic or semantic resources. SOFTCARDINALITY (Jimenez et al., 2013) introduces the eponym measure of *soft cardinality* of a set of elements. Compared to classical set cardinality, this measure computes for each element a weighted inverse of its overall similarity with every element in the set. This means that an element is more valued in this measure if it differs greatly from the other elements, which makes sense when talking about cardinality. This measure leaves the user free to decide the weighting function and the similarity function. The system uses both a similarity on words (with characters as the elements) and a similarity on sentences (with words as the elements). In the case of the soft cardinality of sentences, the weights are weights derived from the cardinality over the words. The system extracts features based on the soft cardinality of various intersections of question, student answer and reference answer and learns decision tree classifiers. Interestingly enough, SOFTCARDINALITY (SC) does not use sequentiality in the sentence, but sequentiality of characters in the words, to deal with typographical errors in student answers. It also includes as a feature the lexical overlap similarity, a strong baseline provided by the organizers of the task, which does use n-grams. On the other hand, ETS (Heilman and Madnani, 2013) builds a classifier with the lexical overlap baseline, character-level n-gram features and BLEU and PERP as its text similarity features. PERP is an edit-based approach to text

System	2-way	5-way
Baseline	0.690	0.430
SC	**0.715**	0.513
ETS	0.713	**0.519**

Table 2.1: Top overall accuracy results at Semeval 2013 Task 7

similarity. It computes the similarity of sentence pairs by finding sequences of edit operations (e.g., insertions, deletions, substitutions, and shifts) that convert one sentence in a pair to the other. Then, using various features of the edits and weights for those features learned from labeled sentence pairs, it assigns a similarity score [1]. No syntactic parsing is needed and no external knowledge resources is used. The two systems slightly improve the overall accuracy of the lexical overlap baseline on the 2-way task. They improve it greatly on the 5-way task, as can be seen in Table 2.1. It is notable that both use character-level n-gram features and greatly improve their performance doing so: on domain-specific question and answers such as these, given that one cannot very well look up words in traditional lexical resources, it is important to be able to closely match two similar-looking words, taking into account variations and typographical errors. Features of this level are to our knowledge specific to this dataset as far as inference goes, in other datasets, they are rarely used.

Finally, in some systems, and contrary to the norm in NLP, features are secondary in importance and complexity to the machine learning method employed. Bu et al. (2012) introduces a kernel method to solve paraphrase identification and RTE. The kernel function operates on two pairs of sentences, and aims at evaluating how similar a pair of sentences is to another. To do that, it counts pairs of n-grams (as a kind of small n-gram rewriting rule) that occur in both pairs of sentences. The features are very simple, yet, this method is able to capture the structure of the sentences and the alignment between entities, obtaining state-of-the-art results on the MSRP paraphrase identification corpus (MRR of 0.76). We will come back to this kind of methods in Section 5, when we present our own string re-writing kernel.

[1] For more information on edit-based approaches, refer to Section 5, which presents our own contribution of this type

2.2.2 Leveraging knowledge on words

As explained in Section 1.3.2.1, to solve a textual inference task, it is often required to bridge the gap between two different formulations of the same concept. The previously described methods can be very effective, but they can also most of the time be improved by using external knowledge resources. In this section, we present lexical resources used for textual inference and a few methods which combine them with a sequential model of the sentence.

2.2.2.1 Pre-processing tools

Every Natural Language Processing system calls on some pre-processing tools to handle raw data. We view them as a kind of knowledge resource. Their use obviously differs from knowledge resources that we can typically look up, but they nonetheless often add to our data a layer of linguistic information drawn from other sources. A *tokenizer* is needed to delimit words and sentences. A *Part-Of-Speech (POS) tagger* annotates the words in the sentence with their grammatical category, offering a first glimpse of non-sequential structure. A *lemmatizer*, which annotates the words with their lemma, or a *stemmer*, for their stems, are a good way to abstract words from their inflections and derivations. One may want to remove words that do not carry sense (like articles), called *stop-words*, and this is usually done in the most basic way by filtering with a manually constituted list.

Most methods we presented previously most likely call on a mandatory pre-processing step as well, but they could very well be implemented with minimal performance loss with a white-space tokenizer and a list of stop-words, so we deemed interesting to separate them from methods which truly rely on complex lexical knowledge.

2.2.2.2 WordNet

WordNet (Miller, 1995) is without a doubt the most popular lexical-semantic database. It groups English words into sets of synonyms called *synsets*, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members. As of 2012, WordNet contains 155,287 words organized in 117,659 synsets for a total of 206,941 word-sense pairs. In modern NLP systems, especially when dealing with inference, WordNet has become the *de facto* resource to include, and it is often interesting to compare the system with it and without it, to evaluate the performance gain brought by *lexical-semantics* resources – WordNet standing for their representative. In earlier pre-

RTE work, WordNet has been found to help passage ranking in question answering: in the evaluation conducted by Tellex et al. in 2003 on the TREC-10 dataset, scoring functions which use WordNet synonyms as the primary query expansion method do consistently better than pure surface form methods (Tellex et al., 2003).

In RTE and more recent inference tasks however, WordNet becomes an even richer resource. In query expansion and paraphrase identification, the equivalence of sense is a strong condition to using a word in place of another, so *synonyms* are often the only WordNet relation worth using without great risk of introducing noise. Textual entailment asks for a one-way relation between sentences, which logically allows for one-way relations between words of these sentences. WordNet contains a lot of interesting one-way relations. On nouns, *hypernyms/hyponyms* allow for a gain or loss of generality, and *meronyms/holonyms* capture which is a part of what. On verbs, *hypernyms/troponyms* also exist for gain/loss of generality, but more importantly, the *entailment* relation, with the same meaning as in "textual entailment". Early after the introduction of RTE, Corley and Mihalcea (2005) propose a knowledge-based method for measuring the semantic similarity of texts. They use word-to-word similarity metrics applied on WordNet. Defined on taxonomies in general, these metrics are still being used nowadays and are worth presenting here. The *Leacock & Chodorow* measure (Leacock and Chodorow, 1998) is determined as in Equation 2.6, with $length$ the length of the shortest path between two concepts, and D the maximum depth of the taxonomy.

$$Sim_{lch} = -log\frac{length}{2 \times D} \tag{2.6}$$

The *Lesk* similarity of two concepts is defined as the word overlap between the corresponding definitions, as provided by a dictionary (Lesk, 1986). In WordNet, the definitions are the *glosses* of each synset. The last measures all make use of the least common subsumer (LCS) of two concepts, i.e. their least common ancestor in the hypernym hierarchy in WordNet. *Wu and Palmer* normalizes the depth of the LCS with the sum of depths of the two concepts (Wu and Palmer, 1994). *Resnik* returns the information content (IC) of the LCS (Resnik, 1995), with the IC defined as $IC(c) = -logP(c)$ where $P(c)$ is the probability of encountering the concept c in a large corpus. Finally, *Lin* adds to Resnik's measure a normalization factor consisting of the IC of the two concepts (Lin and Hovy, 2003). Corley and Mihalcea (2005) combine via a weighted sum the word-to-word similarities to compute the text-to-text similarity. The weights are the Inverse Doc-

ument Frequency (IDF) of the words [2]. They evaluate each similarity measure on the MSRP corpus in paraphrase identification (Dolan et al., 2004), and on the first RTE dataset (Dagan et al., 2006). The final decision is taken by comparing the text-to-text similarity with a threshold learned on training data. They compare with a lexical overlap baseline: the normalized number of common words. On both datasets, the accuracy produced by each similarity measure alone is significantly better than that of the lexical overlap, using a paired t-test ($p < 0.001$). They also combine with a voting perceptron all measures, and this obtains the best results on both datasets: an accuracy of 0.688, for only 0.661 for the baseline on MSRP[3], and 0.583 against 0.545 on RTE. We detailed this early contribution because we deem it foundational for how WordNet and corpus statistics are generally used today to solve textual inference tasks like paraphrase identification and RTE.

In 2006, for the second RTE challenge, Bar-Haim et al. (2006) note that most participants use some form of lexical resource, in particular WordNet. But most well-performing systems also use the structure of a sentence as a tree or a graph, often obtained through dependency parsing. From this point on, most systems will use that structured model of the sentence, and this is the subject of our next part. But changing the language model is no small leap, and we next highlight several recent contributions which do not assume any syntactic layer to the sentence, yet obtain state-of-the-art results on textual inference datasets, proving that a system can still be competitive in current NLP while keeping to the Tier 1 methods of our hierarchy of inference (Section 1.3.2).

Let us go back to Madnani et al. (2012), who use machine translation metrics as features for a classifier on paraphrase identification. As already mentioned, classic translation metrics based purely on surface n-grams do relatively well. But they also use modern machine translation measures that include knowledge resources. TERp (TER-Plus) (Snover et al., 2009) builds upon the core TER algorithm by providing additional edit operations based on stemming, synonymy and paraphrase. METEOR (Denkowski and Lavie, 2010) uses a combination of both precision and recall unlike BLEU which focuses on precision, and incorporates stemming, synonymy with WordNet and paraphrase. Paraphrase is handled by those 2 measures using phrasal paraphrasing on strings (and not on parse trees). While it is difficult to conclude that the inclusion of lexical resources is the main

[2] *IDF*, as an important metric in information retrieval, will be presented more in detail in the next part

[3] As a reminder, a simple system which answers "yes" to all paraphrase candidates gets an accuracy of 0.665

reason of the TERp and METEOR's success, it remains that these two measures are the top-performing individual metrics when used alone as features, with respectively 74.3% and 73.1% of accuracy. When added to the three basic metrics mentioned earlier, the combined features yield an accuracy of 76.6%. With other translation metrics that may be checked out in their article, the complete system reaches 77.4%.

One of the most crucial problems in textual entailment is that of alignment: linking various parts of the Text to their semantic equivalent in the Hypothesis. In fact, this problem is so important that it has spawned the subtask of *monolingual alignment*. The task has been evaluated on the same corpus as RTE, with each text-hypothesis pair additionally annotated with alignments by MSR (Brockett and Dolan, 2005; Brockett, 2007). Figure 2.1 shows an example of gold standard alignment. In recent years, competitive aligners who do not depend on the syn-

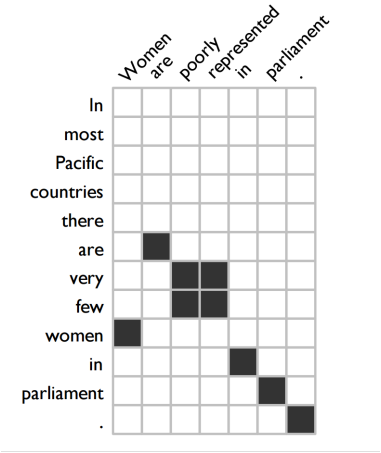

Figure 2.1: Gold standard alignment in the MSR corpus

tactic view of language have been designed. The MANLI aligner (MacCartney et al., 2008) is one of the first monolingual aligners to use the MSR dataset. At training time, sentences are represented as the set of its subsequences. An encoding procedure based on edit operations produces alignments and finally, a scoring function using several features is defined on the edits to score the alignment. The feature set includes the edit type, lexical similarity using aforementioned WordNet metrics like *Lin*, and context (the neighborhood of words). At testing time, the decoding of the best alignment for an unseen sentence pair is computed with a simulated annealing-like procedure. The system beats every known baselines at the

time in term of averages of correct alignments. In recent years, the JacanaAlign aligner (Yao et al., 2013b) uses a CRF model (a machine learning graphical model designed for sequence prediction (Lafferty et al., 2001)) combined with a rich feature set including string similarity (with measures like the Jaccard index and n-gram overlapping), POS tag features, positional features (which evaluate the difference in position from one word with another aligned with it) and WordNet features (using basically all previously introduced relations), but no syntactic features. It outperforms by 66% the reported exact match rate – every alignment in the text-hypothesis pair is correct – of the MANLI aligner (35.3% to 21.3%). They further improve their results on full textual entailment (Yao et al., 2013c) by using a semi-Markov CRF model, an extension of their previous work which allows for phrasal alignment, and not simple word-to-word alignment like in JacanaAlign.

Question Answering has also benefited from extensive use of lexical-semantic resources with no syntactic structure. Yih et al. (2013) in 2013 present lexical-semantic models for the answer sentence selection problem. WordNet is one of their sources of hypernymy and hyponymy. The MRR of their system only including word matching, lemma matching and WordNet reaches 0.715, to be compared with the word counting baseline at 0.652. They complete the system using a variety of automatically built resources, which we will explore later, to remedy several well-known issues. Indeed, WordNet has a rather limited or skewed concept distribution and lacks in coverage of the "IS-A" relation (Song et al., 2011).

2.2.2.3 Other man-made lexical resources

Manually created resources other than WordNet have of course been used in the past. NOMLEX (Meyers et al., 1998) and CATVAR (Habash and Dorr, 2003) provide nominalizations of verbs and other categorial variations of words (e.g., "to invent" and "invention"). It is not worth it nowadays to want to manually create new lexical resources for English. Automatic construction methods are too appealing in term of cost-effectiveness; WordNet is so far ahead in quality and quantity and so standard as a lexical-semantic database that a newly manual resource would not see much use.

However, collaborative lexical resources, like the Wiktionary (Wiktionary, 2002), are promising. As they build slowly over time through volunteers, their coverage becomes remarkable: 618,859 entries for the English language in the English Wiktionary as of 2015. The Wiktionary contain entries with senses and definitions of the word, and can even be parsed for relations like synonymy, antonymy, hypernymy and hyponymy. Some issues remain, like the lack of consistency in

the format of the definitions – despite the existence of guidelines – and the uneven quality of dictionary entries, depending on their popularity and the people who edit them.

2.2.2.4 Drawing on large corpora

As much as it is possible to manually build quality lexical resources, the modern trend in NLP is to build them automatically from a large corpus of text. It is a much cheaper alternative and has become increasingly effective with advances in machine learning and especially the volume and diversity increase of available data.

Simple statistics have been since long computed from large corpora to better model word usage in a language. For example, *tf-idf* is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is defined as the product of term frequency – the number of times the word occurs – and inverse document frequency (Sparck Jones, 1972), defined in Equation 2.7.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.7}$$

where t is a term (a word), N the number of document in the corpus D, and $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears. IDF is a measure of word specificity and as such, is useful for weighting words in information retrieval tasks (Salton and Buckley, 1988). The word counting baseline used by Yih et al. (2013) for answer sentence selection, presented in Section 2.2.1, improves from 0.627 to 0.652 MRR when the words are weighted with their IDF value – considering each candidate sentence as a "document" and the "corpus" as the set of candidates for the same question. Other ranking functions inspired by tf-idf have been later introduced and used as the state-of-the-art in document retrieval, such as Okapi BM25 (Robertson et al., 1995), successful at TREC-3.

A large amount of automatically built lexical resources is founded on the *distributional hypothesis*, the idea that linguistic items with similar distributions have similar meanings, or more simply, that words that are used and occur in the same contexts tend to carry similar meanings. The earlier trend was about distributional thesauri, thesauri generated automatically from a corpus by precisely using this hypothesis (Lin, 1998; Moore, 2001; Brockett and Dolan, 2005; Claveau et al., 2014). But we found that they did not find much success in the past RTE evaluations: Mirkin et al. (2009a) reports in 2009 that Lin's distributional thesaurus has the less impact out of a multitude of other resources, like WordNet and WikiFS

(Wiki First-sentence, a lexical resource built from first sentences in wikipedia articles used as "IS-A" relations). Their other article in 2009 further evaluates the inferential utility of lexical-semantic resources (Mirkin et al., 2009b). They look for relations of lexical entailment from one word to another. Although Lin's thesaurus has better coverage than WordNet, its precision is a lot lower: it provides pairs of contextually similar words, of which many have non-entailing relationships.

It makes sense that distributional resources should not be expected to effectively characterize highly directional and precise semantic relations like entailment. However, the current trend of word embeddings has found successful use in recent work on high-level textual inference tasks, like paraphrase identification, answer sentence selection and RTE. They are a kind of word vector space models, which are surveyed in (Turney et al., 2010). With the resurgence of neural networks and the growing popularity of *deep learning* in the NLP community, several methods have produced word vectors of low dimension out of large corpora, like word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). Without worrying about the technical details – which are outside the scope of this part –, users of these models can simply view these vectors as lexical resources. Neural networks are used to learn a vectorial representation of words with real values, which can be compared using measures like cosine-similarity – which is just computing the cosine of the angle between the vectors. In answer sentence selection, Yih et al. (2013) uses the word vectors from (Mikolov et al., 2010), with a dimension of 640, which helps them reach a MRR of 0.790, their best score and the current state-of-the-art on this task, improving on the MRR of 0.715 obtained by their system using only WordNet as additional resource.

On the same task of answer sentence selection, a kind of recurrent neural networks newly popular in NLP called *long short-term memory* (LSTM) network has been used to reach the best performance to date (Wang and Nyberg, 2015). As all recurrent neural networks, the goal is to go from word vectors to sentence vectors of the same dimension, with the hope that the vector representing the sentence is a semantically close combination of its words. In this case, no dependency parsing is used, and the question and candidate answer sentence are simply concatenated and fed to the network. LSTM has the nice property of dealing with the vanishing gradient effect (Hochreiter et al., 2001) by creating more non-linear connections in the nodes of the network. In particular, this allows in the question-answer concatenation to create a dependency of the answer to the question and the question to the answer. This model uses the vectors in word2vec (Mikolov et al., 2013) and only performs extra processing for numbers and proper nouns, obtaining an MRR

of 0.791.

2.2.3 Conclusion

The first approaches we intuitively think of when solving linguistic problems operate at the word level, which is the Tier 1 of our hierarchy of inferences. They can use different language models, like bag-of-words or sequence of words, and also different external lexical resources. At its most basic, a unigram overlap on surface forms can be computed, but this approach is no longer competitive: it lacks a way to know how the sentence is organized, and the presence of any lexical variation destroys its chances of answering correctly. Doing as little as using n-gram overlaps metrics improves a lot what the method can capture, and is still used to today. Ultimately, on datasets with a lot of lexical variation, systems need to integrate some form of lexical knowledge, which can be found in external resources.

In the next section, we move up a Tier in the hierarchy and examine approaches founded on a structured view of sentences.

2.3 Structural approaches

Natural language, on the surface, has a sequential nature. A word is pronounced, written, read or heard, and then another. This probably stems from the nature of our primary physical channel of communication. Our voice is naturally monophonic and time is for all intents and purposes a continuous stream, so spoken words come out as a sequential signal. Then, written language followed, and while having theoretically more dimensions to work with, was based mostly on spoken language. It is interesting that our most intuitive symbols are initially pictures – be it for a small child or prehistoric people –, quite different from writing.

Nevertheless, it is obvious that there should be an underlying structure to language, different from that of a sequence, because language expresses ideas, and ideas do not seem themselves obviously sequential. In this section, we examine approaches to solve textual inference problems that use structural properties of written language. There are several views of structure that can be adopted, with several different related tools and algorithms. Most have been used in some fashion to solve textual inference problems, but some are of course dominant, like syntactic trees or graphs.

2.3.1 Syntactic dependencies

Using the syntactic structure of a sentence is one of the most common traits of modern NLP approaches. *Syntax* can imply *constituent trees* or *dependency graphs*, obtained by using a parser at pre-processing time. Dependency graphs in particular are preponderant in textual inference, because they allow a straightforward access to relations between words of the sentence without worrying about the distinction between terminal and non-terminal nodes.

We identify five main methods which are based on syntactic features: dependencies as added features, tree-edit methods, tree kernels, latent alignments and recurrent neural networks.

2.3.1.1 Syntax as additional features

Early work with dependency relations applied to textual inference tasks used them mostly as additional features for the same machine learning methods or similarity measures used in previously mentioned contributions. In other words, in this section, the tree or graph structure is not used as the basis of new methods, or a fundamental guide in algorithms, but syntactic features are simply extracted alongside lexical features to enrich the model.

Cui et al. (2005) focus on the benefits of using dependency paths to solve passage ranking on TREC questions. Where others had tries strict matching of relations, they perform fuzzy relation matching, with the idea that two sentences can be semantically equivalent even if the dependencies between entities are not exactly the same. For example, the sentence "Tennis player Jennifer Capriati is 23" answers the question "What sport does Jennifer Capriati play?" even if "Jennifer Capriati" does not have the same dependency relation at all with "Tennis player" as with "play sport". Cui et al. extract pairs of paths, one from the question, and one from the candidate passage, linking the same two words. Then a matching score is computed, considering every alignment of relations from the paths, and using a probabilistic relation-to-relation similarity model. The probabilities are set from two different sources, one with mutual information, and one with weights trained with GIZA. They improve in MRR over three baselines which do not make use of dependencies, by at least 55%. Later, Aktolga et al. (2011) improves that passage ranking method by taking into account the type of the answer and named entities.

In textual entailment, De Marneffe et al. (2006) use dependency graphs as pure representation of the text and the hypothesis. In the first step, their system

computes the best alignment between the graphs. Their hand-crafted scoring measure is designed to favor alignments which align semantically similar subgraphs, irrespective of polarity. It assigns the highest score to pairs of semantically similar words, like synonyms and antonyms. They take into account lemma and POS tag, and use several resources, like WordNet and LSA matrices. They also locally favor alignments which preserve dependency edges. A beam search is used to build the alignments, using the scoring function as heuristic. A logistic regression classifier is then trained with several features on the dependencies and the best alignment, including: polarity, antonymy, adjunct, modality, factuality, numbers, structure, and if the alignment is good or bad based on manually defined thresholds. They evaluate their systems on the RTE 2 dataset, and report through ablation tests that alignment and structure are the most impactful features of their system by far.

In paraphrase identification, Wan et al. (2006) use features like n-gram overlap, as described in the previous section, difference in lengths of the sentences and then add relation overlap features. The relation only takes into account the pair of linked words, and not the label carrying its syntactic-semantic role. They test several learning algorithms, with support vector models performing the best. Their system improves the accuracy significantly over lexical overlap baselines on the MSR Paraphrase Corpus. It is also noted that they include two simple tree-edit distance features. Tree-edit methods are a major class of methods all across the different types of textual inference tasks and are the topic of the next part.

2.3.1.2 Tree-edit methods

Dependency features are not straight-forward to include effectively as features. Recent approaches generally avoid simply adding them to their model, which is why the previous paragraphs did not mention recent work. Care has to be taken to really improve on lexical methods, especially when some recent approaches are successful without using any explicit structural features. One of the most consistent way of using dependencies in past years has been *tree-edit* methods. Their goal is to characterize a sequence of transformations applied to one dependency tree/graph to obtain another. Those transformations are called *edit* operations, or simply edits. They always include at least the insertion of a node, the deletion of a node and the substitution of one node by another. These three edits are shown in Figure 2.2. The first tree-edit algorithms date back to Tai (1979). Basically, different edits are iteratively applied to the source tree, modifying it each time, so that the edited tree becomes closer to the target tree. Eventually, an *edit sequence* is

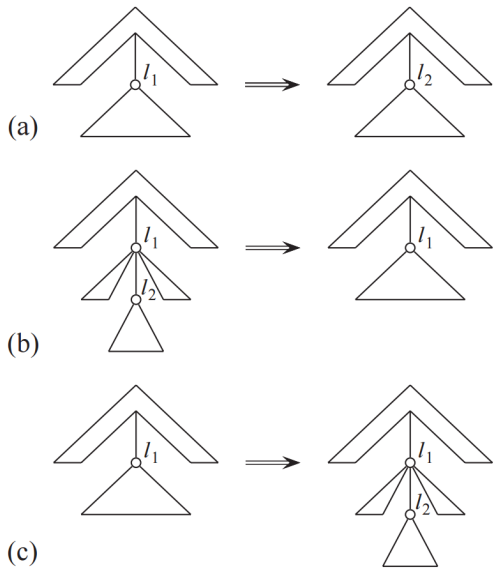

Figure 2.2: (a) Substituting node l_2 to node l_1. (b) Deleting node l_2. (c) Inserting node l_2 as child of node l_1. (Bille, 2005)

found that completely turns one tree into the other and further methods then look at the nature of edits that were effectively applied. They try to characterize if they are elements of proof that the first sentence is in some kind of semantic relation with the other (equivalence, relatedness or entailment), or if the edit sequence is too long or too far-fetched to conclude anything. The exact relation characterized depends on the definition of edits and the definition of their cost or the features extracted from the resulting edit sequence. For example, in paraphrase identification, we might want to prioritize edits which turns a word into a synonym over any other relation like hypernyms or meronyms. In textual entailment, all the relations will probably be more evenly important.

One of the first methods to use tree edits in a high-level NLP task is that of Punyakanok et al. (2004), with a contribution on question answering on TREC data. Trying to transform the question sentence into the full answer sentence does not really correspond to the task of finding and extracting the answer. That is why their edit distance is specifically designed to allow entire subtrees to be deleted at once. They report an accuracy on correct answer passages of 36.6% compared to their bag-of-words baseline at 26.2%.

Several recent tree-edit methods are able to integrate complex edit operations

and sometimes lexical resources. Heilman and Smith (2010) employ a tree kernel[4] as a heuristic in a greedy search routine in the search space of edited trees, to efficiently extract sequences of edits. Their extended edit set includes moving a subtree from one parent node to another, and also moving siblings, to shorten the edit sequences. They extract features from the obtained edit sequences – mostly counts of edits, with separate features for type of edits, words, lemma, POS tags, proper nouns – and use them in a logistic regression learning model. Interestingly, they evaluate their method on three tasks: paraphrase identification on the MSRP corpus, textual entailment on RTE 3 (using RTE 1 and 2 as training) and answer sentence selection, on Wang's TREC dataset (Wang et al., 2007). RTE 1,2,3 are the most freely available RTE datasets, so are still very standard in today's RTE research. Table 2.2 (p. 81) presents the results of the contributions described in this literature review. Likewise, Wang's dataset has become standard in the sentence selection part of today's small-scale QA research. Table 2.2 (p. 81) also compiles the results of contributions in this literature review. Heilman and Smith obtain competitive results in term of recall on paraphrase identification, textual entailment, and the best results at the time on the question answering task in both MAP and MRR, without using WordNet like the previous systems, which is remarkable. The results table include the three tasks, so that it is easier to assess the quality of systems evaluated on several tasks, compared to systems only dedicated to one.

The previous method requires a separate alignment-finding phase and resorts to ad-hoc distance metrics. Unlike Heilman and Smith (2010), Wang and Manning (2010) treats alignments as structured latent variables, and learn both the alignments and the final decision in the RTE and answer sentence selection tasks. At the core of the model is a tree-edit algorithm. As systems dealing with latent alignments are the topic of Section 2.3.1.4, we will detail their system in this part.

(Yao et al., 2013a), working on question answering, propose the use of a CRF (Lafferty et al., 2001) in order to cast the problem as one of sequence tagging by labeling each token in a candidate sentence as either Beginning, Inside or Outside (BIO) of an answer. They use traditional contextual features based on POS tagging, dependency parsing and named entities, but also features from tree edit distance model for aligning an answer sentence tree with the question tree. In that respect, this system is pretty similar to Wang and Manning's system, but it is capable to extract the exact answer, and not just select the sentence containing it. They additionally use n-gram context features and question type features. Interestingly,

[4] A function that measures the similarity of two trees, as we will see in the next part

their system is only trained on positive examples (sentences with the correct answer) because otherwise the system became too reluctant to label an answer. They obtain state-of-the-art results on answer sentence selection (Table 2.2 p. 81) and report a gain of 0.02 MRR with the use of WordNet in the edit operations.

Tree edits are very convenient for specifying the kind of transformations we expect to encounter between two sentences semantically related. Although this formalism is very flexible, it has some limitations. First, it proceeds most of the time in two steps: edit applications, and then some kind of cost computation or feature extraction. It is difficult to integrate those two parts, especially in a coherent machine learning method. Second, it is difficult to know what features are interesting after computing the edit sequences, because the meaning of the final edit sequences is quite unpredictable: a delete edit could mean that the word is superfluous, or that it changes the sense completely. Finally, tree edit methods often feature a lot of hyper-parameters to fine-tune. Values of these parameters might not be transferable to other tasks.

The next major class of methods does not explicitly extract features for each sentence pair or text-hypothesis pair. Instead, they rely on computing a similarity function between two instances of the problem, that is to say, between two pairs of two sentences, hence involving 4 dependency graphs.

2.3.1.3 Tree kernels

Kernel functions measure the similarity between two elements. Used in machine learning methods like SVM, they allow complex decision functions to be learned in classification tasks Vapnik and Vapnik (1998). The goal of a well-designed kernel function is to have a high value when computed on two instances of same label, and a low value for two instances of different label. We will be coming back to kernel methods later in Chapter 5 to present our own contributions. For now it is enough to know that the crux of the next contributions is in the design of a function capable of assigning a real value to a *pair of pairs of sentences* – as we are working on the similarity between two text-hypothesis instances. These methods are useful when features for an individual instance are hard to formulate and especially when they are computational expensive. We do not care about the whole of 100,000 features of a text-hypothesis pair A when it only has in common 10 values with another similar text-hypothesis pair B. Kernel functions allow to only consider what is similar or different in two instances and not what a single instance represents.

Zanzotto et al. (2007) use a kernel method on pairs of syntactic trees, inspired by their previous work introducing a *cross-pair similarity* on text-hypothesis tree pairs. They later expand on graph kernels in a second method (Zanzotto et al., 2010). Their method first aligns tree nodes of a pair of sentences to form a single tree with placeholders. The placeholders are linked to anchors attached to word nodes, so that placeholders link semantically equivalent words. Figure 2.3 shows a text-hypothesis pair with numbered anchors on equivalent words. Figure 2.4 shows the resulting tripartite graph obtained by linking 2 graphs via placeholder nodes. They then use a simple tree kernel (Moschitti, 2006) to compute the number of common subtrees of two of those tripartite trees (representing a T-H pair). A SVM is used to finally classify T-H pairs. One of the weaknesses of the method is that they are forced to align the sentences as a separate first step, which is usually done with ad-hoc methods. To get a better matching, they use WordNet relations and a WordNet similarity metric similar to the ones described in Section 2.2.2 (Jiang and Conrath, 1997). Their method obtains the best accuracy to date on RTE 3 (Table 2.2 p. 81), out of all systems using only dependencies and WordNet.

Severyn and Moschitti (2013) build upon this work to design a very similar system, but for question answering. This system leverages additional resources and features, like question classification, lexical answer types, and named entities. They get the best results at the time in MAP and MRR on Wang's answer sentence selection dataset (Table 2.2 p. 81), and also report that answer typing with named entities improve their results considerably: they do not get the best results without it.

To conclude this section, we briefly present the very recent work of Filice et al. (2015). They provide detailed evaluation of a multitude of such tree and graph kernel methods. While the background algorithms are pretty much the same as Zanzotto's previous contributions (adding only the kernel defined by Costa and De Grave (2010), they add many more lexical-semantic resources and similarity measures as features, including: Linked Open Data and WordNet, syntactic and kernel similarities, longest common subsequence and other string similarity measures, Resnik similarity, Wikipedia, Wiktionary. The combination of multiple kernels yields the best result to date on the MSR Paraphrase corpus (Table 2.2 p. 81), and they report their best competitive results on RTE 3 using the sole *Smoothed Partial Tree Kernel* (Croce et al., 2011).

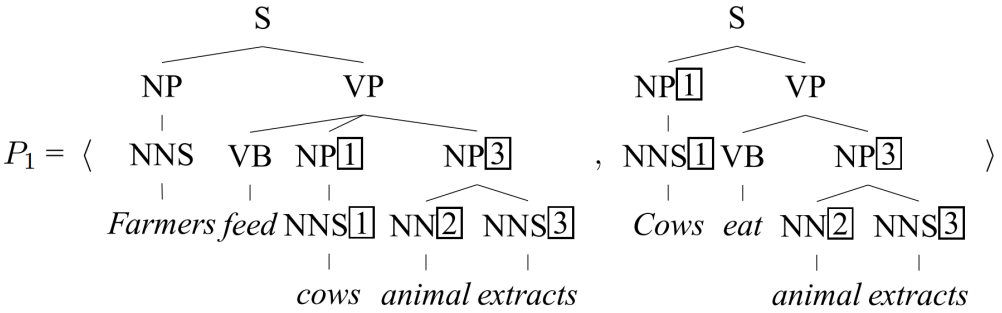

Figure 2.3: A pair of text and hypothesis trees, with anchors

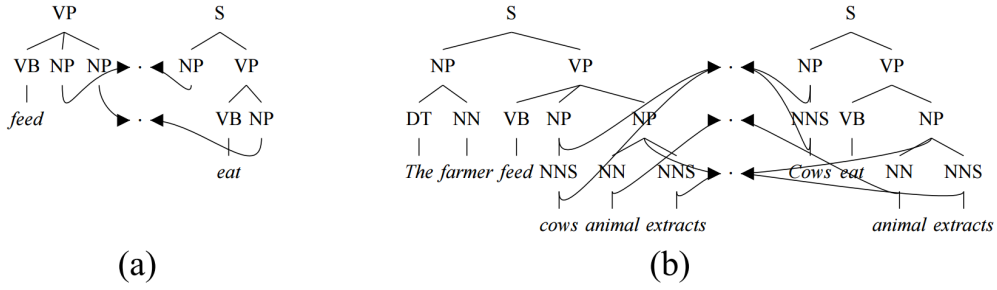

Figure 2.4: (b) A tripartite graph, showing placeholders

2.3.1.4 Latent alignments

The previous syntactic methods dealing with alignments do not propose an integrated process for both learning the alignments and the decision to the inference problem. For example, Zanzotto et al. (2010) are designed to compute an alignment *a priori*, using heuristics, and only then learn the textual inference judgment assuming that alignment. The following methods implement *structured prediction* by taking into account in the learning model the *hidden* or *latent* alignments from one sentence to the other. This aims at being more robust under unpredicted phenomena and pre-processing errors.

Probabilistic quasi-synchronous grammars have been used with some success (Wang et al., 2007; Das and Smith, 2009). This class of grammars generates both a source sentence and a target sentence according to probabilistic rules akin to those in probabilistic context-free grammars. It allows to compute probabilities of one sentence tree being generated knowing another sentence tree. Probability computations include conditional distributions over parts-of-speech, named-entity labels, dependency relation labels, and mixed with probabilities involving WordNet relations. Finally, a log-linear model learns those parameters. Wang et al. (2007) use this model to select answer sentences to questions from TREC data, already mentioned in this section. They outperform previous passage retrieval methods like the one of Cui et al. (2005) and report a significant improvement when using their mixture model with WordNet (Table 2.2 p. 81). Das and Smith (2009) use a similar model for paraphrase identification, adapting the original model by Wang et al. from question-answer pairs to paraphrase pairs. They also add a *product of expert* to include another posterior probability estimate based on lexical overlap. Evaluating on the MSRP corpus, they report their best results when using this method over only using the quasi-synchronous grammar or the lexical overlap probability estimates (Table 2.2 p. 81) and outperform previous work.

Wang and Manning (2010) use probabilistic finite state machines to model the transformations from one sentence to another in textual entailment and question answering. They use a rich edit operation set that takes into account relations in WordNet (editions of synonym, hypernym, antonym), if the edited word is a modal verb, a word that expresses likelihood (like “maybe”, or “possibly”) or a named entity. Their model is a tree-edit CRF (Lafferty et al., 2001) with a finite-state machine-based method for mapping edit sequences to optimal weights. Their features include internal information of their model, like the state of their FSM, and also typical linguistic features, like word matching and tree structure. They obtain competitive results with comparable systems on RTE 3 and the best results

at the time on the QA task in MRR (Table 2.2 p. 81). There is a duality between probabilistic FSMs and probabilistic formal languages, which they used in their previous work (Wang et al., 2007). Even if their more recent contribution models transformations and their previous one models sentence pair generation, using otherwise different classes of formal languages, it is still interesting to observe that latent alignments can be captured using formal languages designed specifically for sentence pairs.

Finally, the recent contribution of Yih et al. (2013) propose one model based on latent syntactic alignments, comparing it with their aforementioned lexical methods. In order to leverage the latent alignments between question and answer, they adapt the framework of *learning constrained latent representations* (LCLR) proposed by Chang et al. (2010). Without going into too much detail, this method involves accounting for structure as constraints of an Integer Linear Programming problem. The coefficients of the solution of the ILP formulation are added to the objective function of the learning method, which effectively allows to iteratively both learn the decision to a classification problem and the latent structure of the sentences. All of Yih et al.'s models are enriched with the same lexical-semantic resources, and they report better results in term of MAP for LCLR, but worse results in term of MRR, than their bag-of-word model.

2.3.1.5 Recurrent neural networks

Recurrent neural networks are the latest trend in NLP and have been recently adapted for use throughout most NLP applications. Their effectiveness remains to be proven on complex textual inference like machine comprehension but nevertheless has already been shown on simple two-sentence inference problems. We already presented the work of Wang and Nyberg (2015) on LSTM networks for question answering. This model is used to capture the relation between words of the question and words of the answer sentence candidate, but they only use a sequential chaining of the words, and not their syntactic structure. The idea is that LSTM may be powerful enough to capture sufficient context and solve a sentence retrieval task on factoid questions.

Socher et al. (2011) propose a recursive auto-encoder to learn a vector representation of a sentence given vector representations of its words. Two vectors of children nodes of the parse tree are combined through a matrix operator to obtain a single fixed-sized vector representing their parent. An encoding matrix and a decoding matrix are learned. What is interesting in their method is that this operation can be executed recursively over the tree structure to produce only one

vector representing the whole sentence. In the paraphrase identification problem, sentences are then compared using the cosine-similarity of their vector representations. They learn the initial recursive auto-encoder on 150,000 sentences from the NYT and AP sections of the Gigaword corpus. In the final classifier evaluated on the MSRP corpus, they also add unigram overlap features and features about numbers (as this is generally important to get exact matches for numbers in this dataset) to word-vector features. They report the best results at the time (Table 2.2 p. 81). The recursive auto-encoder alone does worse than the added basic features alone (on the level of 2006 methods), which shows that some information is lost in the encoding and decoding phases, but the combination yields a considerable improvement over both sets of features.

At the time of writing, we do not know of deep learning methods which solve one-way textual inference. The problem is that contrary to paraphrases, general textual inference is not only interested in modeling the sense of one sentence, to compare it with other sentences with close semantic similarity, but is most interested in the relation between a specific pair of sentences, which has not as of yet correctly been modeled by neural network approaches.

Syntactic dependencies are often a way to assess which entities are linked together in the sentence, without having a clear idea of what their semantic role actually is. The next section presents methods that assume a much more semantically significant layout of the sentence, in particular by setting aside the distinction between verb and noun phrases and focus on predicate-argument structures.

2.3.2 Semantic structure

Textual inference is definitely a problem about semantics. Syntax can be seen as only a low-level approximation for semantics, but some contributions also use higher-level representations, in the form of *predicate-argument* structures and with a notion of *semantic roles*. It is worth noting that to our knowledge, these methods are not competitive on our tasks when they choose to completely ignore dependency structure and replace it with semantic roles. The reason is most likely the insufficient precision and coverage performance of tools employed to access the semantic layer of the sentence. Those tools are quickly reviewed in Section 2.3.4.

Qiu et al. (2006) use prototypical predicate-arguments structures in addition to parse trees, in the task of paraphrase identification. Usually following the Prop-

Bank notation (Kingsbury and Palmer, 2002), these annotations are limited to roles like "predicate", "argument 1", "argument 2". This work uses predicate-arguments as information nuggets, to be compared between one another, and aligned. To better perform this matching, they use the distributed thesaurus by Lin (1998). Manual rules determine what is important and what is superfluous in the nugget so that extra information generally does not affect the matching negatively. After the first alignment phase, a classic feature extraction is performed, including features on dependency paths, and predicate or argument mismatches. Their SVM obtains high recall on the MSR paraphrase corpus, and outperforms simple baselines (Table 2.2 p. 81).

Recently, Lien and Kouylekov (2015) tackles textual entailment with a rule-based system, i.e. without machine learning. They use the English Resource Grammar (ERG) (Flickinger, 2000), a general-purpose open-domain semantic parsing system that outputs logical-form representations. This output is naturally translated to a variable-free semantic dependency graphs, with nodes annotated in simple semantic roles like ARG1, ARG2, or BV (bound variable) on words like pronouns. Lien and Kouylekov then produce RDF graphs[5]. The graphs are enriched in a forward-chaining spirit using rules in SWRL format, and the Jena reasoner[6]. The rules include: abstract rules (to match indefinite and personal pronouns in the hypothesis to existing Noun Phrases in the Text), simplification of predicate, structural rules (to handle phenomena like conjunction), filtering to remove superfluous graph nodes. Then the hypothesis is converted to a SPARQL query and the graph is queried that way. Depending on the answer, a final decision process decides between 3 classes: entailment, contradiction and neutral. They report high accuracy on the Parser Evaluation using Textual Entailments (PETE) task (Yuret et al., 2010) and on SemEval 2014 Task 1 (Marelli et al., 2014). Note that we did not mention those datasets before, because the corresponding evaluation challenges initially aimed at evaluating textual entailment in very specific settings: one evaluates parsers, and the other compositional distributional semantic models. As such, in our understanding, those datasets are not yet mainstream in the textual inference field because comparison with the previous methods is difficult.

Semantic information of higher level can also be accessed. Shen and Lapata (2007) use the semantic role annotations defined in Framenet (Baker et al., 1998).

[5]Resource Description Framework (RDF) is a popular format for modeling data in the Semantic Web.

[6]`https://jena.apache.org/`, (Carroll et al., 2004)

They allow a sentence to be annotated like in the following example:

(32) $[\text{Lee}]_{\text{Seller}}$ sold $[\text{a textbook}]_{\text{Goods}}$ $[\text{to Abby}]_{\text{Buyer}}$.

Semantic role labelers with FrameNet annotations were not available at the time, so they instead apply an ad-hoc lookup method to match verbs to FrameNet predicates, and their syntactic dependencies to FrameNet arguments. They then use a pairwise similarity measure to create matchings between these semantic structures, in the same fashion as Qiu et al. work. As they work on TREC questions, they also assign to the question word its corresponding semantic role. If they can find an argument with a matching identical role in the candidate answer sentence, they answer the question by that argument. On the task of question answering, they report a vast improvement in performance when compared to a pure dependency path matching baseline and another baseline using Shalmaneser, a state-of-the-art shallow semantic parser (Erk and Pado, 2006).

Up until this point, we have only explored systems dealing with pairs of single sentences. Sometimes they can process pairs of multi-sentence text snippets – for example, by simply concatenating the sentences – but for the most part, they were not designed to handle multiple sentences. In the next section, we expose several problems arising from multi-sentence text.

2.3.3 Multi-sentence problems

Most of the real-world textual inference problems concern texts with multiple sentences. The problem of RTE, with 2 sentences (in most cases), was created specifically to simplify and have more control on the difficulty of the problem. RTE remain a very hard NLP question and is far from solved, but it is obvious that multiple sentences have to be addressed. A part of research on textual inference is already trying to, with the effort revolving around automatic reading comprehension tests. In this section we focus on two important types of processing: coreference resolution and discourse relations parsing.

2.3.3.1 Coreference resolution

There are several kinds of coreference: anaphora, cataphora, split antecedents and coreferring noun phrases (Jurafsky and Martin, 2000). No matter the type, the non-trivial problem is always to bind two entities, given that one is generally well-defined and the other is semantically vague or ambiguous, like a pronoun.

The need for coreference resolution can happen within the same sentence, but in general ignores sentence boundaries.

In reading comprehension tests, questions are asked on a full text, with more than 50 sentences, most of the time narrative. There is a fixed set of characters, and their names are not used in every sentence they appear. Coreference resolution is hence one of the most important cross-sentence phenomenon to take into account when solving this automatically. At CLEF 2014 in the Entrance Exams task, Peñas et al. (2014) report that 4 out of 5 participants use coreference resolution. The best system in 2014 and 2015 is that of Synapse Développement (Laurent et al., 2014, 2015). Their anaphora resolution step considers personal pronouns ("I", "me", "myself"), demonstrative pronouns and adjectives ("this" and "that"), possessive pronouns ("my", "mine"), relative pronouns ("who", "whom", "whose", "that") and the pronouns "one" and "ones". During parsing, the system builds a table with all possible referents for anaphora ((proper nouns, common nouns, phrases, clauses, citations) and bind them using features like gender, number, type of named entity, category in their in-house taxonomy, position of the sentence. They report that their method obtains state-of-the-art results when evaluated intrinsically. They use many more in-house resources in their system, like negation, modality, semantic relations like WordNet's, semantic frames a la FrameNet. It is impossible to really know how much anaphora resolution truly contributes to their system, but they still obtain remarkable results on the task, beating the second system 0.58 to 0.36 in accuracy.

Coreference resolution is not really used outside of tasks dealing with full texts. Tasks like paraphrase identification and RTE most often feature instances with only two sentences. This does not justify the risk of adding noise in the global system because of errors in the coreference resolution step.

2.3.3.2 Discourse relations

Discourse parsers have seen very little use in textual entailment tasks so far, probably due to the low coverage of annotations on complex narrative texts. Some tools are based on Rhetorical Structure Theory (RST), which defines relations like Elaboration, Evidence, Condition, Means, Purpose, Unless between two large word spans (most of the time between two complete propositions) (Mann and Thompson, 1988).

Recently, Sachan et al. (2015) tackles machine comprehension, on a new dataset by Microsoft Research, called MCTest (Richardson et al., 2013). MCTest is a set of 660 stories and associated questions intended for research on the ma-

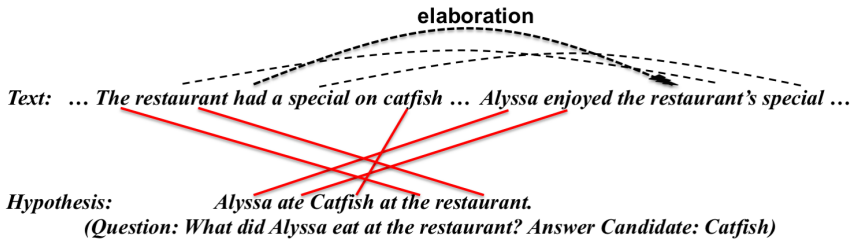

Figure 2.5: Answer-entailing structure for an example from MCTest500 dataset

chine comprehension of text (also called machine reading, cf Chapter 1). The data was gathered using Mechanical Turk. Each short narrative text is provided with 4 questions, each with 4 answer choices. An example of reading comprehension test is available on the next page.
Sachan et al. learn latent *answer-entailing structures*, i.e. the best (latent) alignment between a hypothesis (formed from the question and an answer choice) with appropriate snippets in the text that are required to answer the question. Figure 2.5 shows an example of answer-entailing structure. Their learning process integrates the ranking structure for text-hypothesis alignment to the features, which allows their objective function to capture the alignment optimization. This is called *structured prediction*. They again specialize their objective function by capturing the notion of "multi-tasks", the idea that a complex task such as reading comprehension has to be divided in a variety of subtasks. The technical aspects can be further read in the article. Their features set include: word-level surface-form matching; word similarity for synonymy using word vectors; relations from WordNet; named entities and events; contextual features based on n-grams, dependencies and predicate-argument structures; additional global features like a tree kernel; RST (with the discourse parser HILDA (Feng and Hirst, 2014)) and coreference links; string edit features using paraphrase databases. They note that their model cannot handle negation well (or any kind of polarity), so incorporate a small set of manual heuristic rules to that effect.

Before addressing their experiments, let us present a very recent submission from Weston et al. (2015). They identify a set of 20 elementary subtasks in the problem of complex question-answering and present them in the same way we did in Section 1.3.2, by providing a simple example which cannot be solved without dealing with the related phenomenon. They define subtasks like Factoid QA with a single supporting fact, with more supporting facts, Two argument relations,

Text: James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.
One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.
His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

Questions:

1. What is the name of the trouble making turtle?

   A Fries
   B Pudding
   C James
   D Jane

2. What did James pull off of the shelves in the grocery store?

   A pudding
   B fries
   C food
   D splinters

3. Where did James go after he went to the grocery store?

   A his deck
   B his freezer
   C a fast food restaurant
   D his room

4. What did James do after he ordered the fries?

   A went to the grocery store
   B went home without paying
   C ate them
   D made up his mind to be a better turtle

Yes/No Questions, Counting, Coreference, Deduction/Induction, Spatial reasoning and others, whose complete description can be read in their article.

Sachan et al. (2015) use that classification to annotate the questions in MCTest. This allows their multi-task model to indeed have a different behavior on different types of questions. For their experiments, they use several baselines, and evaluate on answers with single and multiple justification sentences in the text. Interestingly, the "RTE" baseline performs better than their system on single sentences, but its performance drops greatly on multiple sentences. Overall, their multi-sentence multi-task model yields the best overall accuracy, with 67.8%, improving over 59.9% the best previously-published result, obtained using the recursive neural network for factoid question answer of Iyyer et al. (2014).

2.3.4 Knowledge on structure

As with lexical methods, structural methods can be improved by enriching them with knowledge. With pre-processing tools specific to structure, resources at the lexical level, described in Section 2.2.2, are still the most commonly used. This section gives a brief overview of knowledge resources pertaining to how several words are combined, as opposed to only one word.

The following pre-processing tools all extract a particular layout of words expliciting different kinds of relation between them. A *syntactic parser* outputs an analysis of a sentence as a tree of its constituents or a graph of dependencies between words. These dependencies can be seen as a primal form of semantics but are generally of a more purely functional nature. Many parsers include *coreference resolution*, tasked with linking expressions in the text referring to the same entity. They are often limited to simple anaphora and cataphora involving pronouns, but do not handle coreferring noun phrases well: one might say this last linguistic phenomenon is one of the very goals of textual inference. Most NLP toolkits include a state-of-the-art syntactic parser and coreference resolution, like Stanford CoreNLP (Manning et al., 2014), NLTK (Bird et al., 2009) and OpenNLP (Morton et al., 2005). As far as availability is concerned, dependency parsers are generally a bit more difficult to create than simple tokenizers and POS-taggers, and often rely on training data such as a labeled tree bank. As such, effective implementations may not be available for less studied languages. A *Semantic role labeler* provides a more direct access to a semantic layer of the sentence. It annotates a predicate and its arguments in the sentence and assigns roles in relation to each other. SRL is generally less successful than syntactic parsing and less tools are available: Illinois Semantic Role Labeler (Punyakanok

et al., 2008), based on the Propbank annotations, and SEMAFOR (Das et al., 2014) based on Framenet's frame semantics.

A *paraphrase collection* is a natural knowledge resource in textual inference. It contains equivalences between phrases, often in the form of patterns around a verb, with variables as arguments. For example, here are the top five paraphrases for "X solves Y" generated by DIRT (Lin and Pantel, 2001): Y is solved by X, X resolves Y, X finds a solution to Y, X tries to solve Y, X deals with Y. The variables allow for shifts in argument order and position. The more recent PPDB (Ganitkevitch et al., 2013) offers roughly the same format for some 169 million rules for their most inclusive set, and has been created with bilingual parallel corpora. (Hickl et al., 2006) present a rich system obtaining the best result at RTE 3 (Table 2.2 p. 81) and still unrivaled ever since on this dataset. They use a lot of pre-processing tools, resources and sub-modules, but most importantly, they incorporate a method for automatic paraphrase acquisition on a large corpus. They report that the largest gains in accuracy are obtained by adding features based on the paraphrases extracted that way. When used alone, paraphrase features yield 66% in accuracy, which is in the range of the best systems based on lexical-syntactic features with WordNet as of today.

Other resources which could be used in textual inference are automatically and manually created *triplet relation databases*. TextRunner (Yates et al., 2007) and ReVerb (Fader et al., 2011) are automatic methods to identify and extract relations from English sentences, to be used on Web texts. ConceptNet (Havasi et al., 2007) is a relation graph on commonsense knowledge built entirely manually. Finally, databases like Freebase (Bollacker et al., 2008), Yago (Mahdisoltani et al., 2014) and DBpedia (Auer et al., 2007) provide knowledge on named entities, like people, organisations and locations. They are usually built collaboratively, with Freebase being built directly by its community, and the other two databases being extracted from Wikipedia pages. Although this type of resources has recently seen major use with tasks like Question-Answering on Linked Data (Cimiano et al., 2013), their use remains for now limited to factoid question answering and they have not been used in more open types of textual inference problems.

2.4 Conclusion

This literature review presented what has been accomplished so far in textual inference. Systems implement a lot of different methods and are evaluated on a lot of different tasks. We chose the angle of the features they use to organize this chap-

ter. After introducing the evaluation methods, we first focus on lexical approaches, that is to say, the Tier 1 of the taxonomy introduced in Chapter 1. Bag-of-words on surface forms alone are not effective enough, but n-gram overlap methods can still be used today with success. We then explore structural approaches, modeling a more hierarchical arrangement of the sentence than its simple sequentiality. This corresponds to the Tier 2 of the taxonomy. In practice, systems often use syntactical structure to assess the semantics of a sentence, rather than relying directly on tools or methods annotating its semantic structure. On the word level as well as the structure level, methods can integrate external resources to bridge the surface variations and boost their performance.

We note that there was no part corresponding to Tier 3. We believe that a great limitation imposed on current systems is the lack of data to train and evaluate on. The available data is small in size: most of the datasets we mentioned contain a number of instances in the order of a thousand at most. And it is not designed for higher level inference phenomena like those of Tier 3. Admittedly, this data would be very hard to produce at a large scale. The consequence for systems is that first, they must restrict what to try to learn on this amount of data, and second, there is no interest in designing scalable systems in the first place. We believe the research community should devote some effort to the construction of large-scale datasets and large-scale systems. The case of systems has been addressed recently with the Excitement Open Platform (Magnini et al., 2014), an open scalable textual entailment system. It provides the capability of using multiple knowledge resources and textual entailment algorithms, as well as including any new resource or algorithm within the pipeline. Our literature review focused more on methods and features, and mostly ignored these scalability and dataset issues. We did not address them in our contributions either, but they must remain an important consideration for any researcher in this field.

What we conclude from this review for the direction of our own work is that there is no sure-fire way to improve a system by just adding features like syntax or knowledge resources. Tier 1 methods can be as successful as Tier 2 methods, on just about any inference-related task. This is not in contradiction with the classes of our taxonomy: a class represents the lowest level necessary to solve *completely* the task, in an ideal setting. As it stands, inference is still far from being solved by any system, so the performance ceiling of the various approaches is still unknown. Some general principles still seem to hold. Ignoring structure altogether – even a simple n-gram model – is not a viable approach in highly directional and precise tasks like recognizing textual entailment. Also, a lot of contributions seem to draw a significant benefit in the application of external knowledge like WordNet

and paraphrase databases. In the remainder of this dissertation, we build on those two principles and try to integrate them tightly in our systems.

In the next chapter, to detach ourselves from the distinctions between all the tasks related to inference, we propose a more general abstract problem encompassing all the applications described up to this point. We match this problem with a formal proof system solving it, and show that it can shed some light on the capabilities required of systems.

	MSRP		RTE 3	TRECWang2007	
System	Accuracy	F-score	Accuracy	MAP	MRR
All-positive baseline	66.5	79.9	51.2	0.397	0.493
Punyakanok et al. (2004)				0.419	0.494
Cui et al. (2005)				0.435	0.557
IDF word-count				0.596	0.650
Qiu et al. (2006)	72.0	81.6			
Wang et al. (2007)				0.603	0.685
Heilman and Smith (2010)	73.2	81.3	62.8	0.609	0.692
Wan et al. (2006)	75.6	83.0			
Das and Smith (2009)	76.1	82.7			
Wang and Manning (2010)				0.595	0.695
Bu et al. (2012)	76.3	N/A	65.1		
Zanzotto et al. (2007)			65.8		
Socher et al. (2011)	76.8	83.6			
Madnani et al. (2012)	77.4	84.1			
Severyn and Moschitti (2013)				0.678	0.736
Yao et al. (2013a)				0.631	0.748
Yih et al. (2013)				0.709	0.770
TESRK (Chapter 5)	77.2	84.0	66.1	0.672	0.768
Filice et al. (2015)	79.1	85.2	67.0		
Hickl et al. (2006)			80.0		

Table 2.2: Results on MSR Paraphrase (MSRP), RTE 3, and Answer Sentence Selection (TRECWang2007)

Chapter 3

A theoretical model to solve the "Contextually queried inference" task

In the context of this thesis, we tackle multiple different tasks all related in some way to textual inference. In this section, we define a general abstract super-task representing our vision of what most – if not all – these tasks derive from. We argue that this is useful for several reasons. First, such a task would only focus on what we deem *necessary* to the larger textual inference problem, rather than *contingent* to a specific application, like question-answering. We can formalize these principles and set them as the foundation of any system intending to solve inference. Then, a hypothetical system designed for this task would be usable on all the derivative applications and could leverage all the data available for different tasks as one single training corpus. Finally, by doing so, this system would be able to use capabilities traditionally not sought-after for a specific application of textual inference, simply because these capabilities are inherently required for other applications; we believe that incorporating all these capabilities into one single framework could help advance textual inference in all directions of its applicative field.

In this chapter, we first define the general problem of Contextually Queried Inference. Then, we present a proof system providing a formal way of solving CQI. Finally, we introduce the practical contributions of the thesis, detailed in Chapters 4 to 6. We take a look at them through the lens of CQI, that is to say, we present which capabilities of the proof system each of our contributions is designed to capture.

3.1 Contextually queried inference

3.1.1 Definition

Contextually queried inference (CQI) is our proposition for a general textual inference task. To the notion of *inference*, it adds the less commonly associated notions of *context* and *queries*. The classic textual inference task, recognizing textual entailment (RTE), generally deals with on very short text fragments, no longer than two or three sentences, which makes for minimal context, and there is no "query" to speak of. This was, of course, intentional when the task was designed: the goal was to bring a more focused and precise evaluation as well as allow less complex and costly implementations.

A CQI instance is defined by a tuple $\langle Q, K, T, H \rangle$, defined as follows:

- Q is the *query*
- K is the *context*
- T is the *text*
- H is the *hypothesis*

The text and the hypothesis are in textual form and have the same role as in the RTE task. The decision problem is, at its core, "can a human reasonably infer H from T". But this time, the machine reader is given a larger context K, representing some sort of knowledge base in unstructured (textual) form or structured form, to help its decision, for example by looking it up for helpful details about H and T. The query Q is an initial clue that the reader is presented with, separate from K, which generally prompts it to look for a relevant bit of information in K. The complete problem can be phrased as:

"Having retrieved information by querying K with Q, can a human reasonably infer H from T?"

By convention, an empty query means that the entire context K has to be considered. The way the actual information retrieval step is achieved is open-ended. We originally mean it to be done purely by the same human reader, simply reading some words or a question (as Q), and a sizable text (as K), but it would be interesting for other applications to consider how humans draw inferences given the output of a given automatic search engine. The answer can be presented under

several forms, as the original RTE: yes or no, multiple classes, numeric judgment. An interesting variant of the complete problem is to give 3 elements of the CQI tuplet, and ask for a fourth one (among a list of choices) such that the answer to the question is "yes".

Why is adding Q and K interesting? One might argue this is only going back to question-answering and all its quirks and difficulties, something RTE distanced itself from. A big restriction on the current form of RTE is that the two short text fragments – let us assume they are two sentences, which is most of the time true – have to be self-sufficient for the RTE instance to make sense and evaluate automatic systems in a fair way. Consider the following positive instance (from RTE 3 data):

(33) **T:** *She was transferred again to Navy when the American Civil War began, 1861.*
H: *The American Civil War started in 1861.*

The machine does not have to know what the American Civil War is. Now consider the following positive example:

(34) **T:** *It didn't happen because the cream of England's thugs was smoking pot which is easily and legally available in the Netherlands.*
H: *Drugs in Holland are easily bought.*

Here, the machine does have to know that the Netherlands and Holland mean the same thing. If machines are required to know certain things, why not include in the task an indication for where to learn the needed information? Now consider the following negative example:

(35) **T:** *The existing education legislation does not recognise home schooling.*
H: *Reform allows home schooling.*

This example is more tricky. Both existing legislation and a reform are mentioned, and it would make sense that if they are paired with the same topic (home schooling), they behave differently regarding that topic. It would be an interesting reasoning step for a machine to perform, but it is not the correct one in this case: the machine has to decide it has too little information to conclude on that inference. This also is interesting reasoning, but some additional context would allow for an even more focused evaluation in this case, if we could somehow isolate one or the other reasoning idea. CQI is an answer to that need.

In the next section, we present classic textual inference tasks in a CQI form.

3.1.2 Framing classic tasks as contextually queried inference

RTE in its classic form is obviously the simplest task to frame as a contextually queried inference problem. With empty context and query, all that is left in CQI is the original textual entailment problem, the core *inference* part of CQI. Let us examine other tasks.

Paraphrase identification is also straight-forward: it is equivalent to solving two problems with empty context and query, switching roles of T and H.

What about question-answering? Let Q[Wh] be a factoid question containing a question word Wh (this includes noun phrases like "which country"). With K a text or a database, the problem is then to find a passage P[A] in K containing an answer candidate A such that the answer to the CQI problem $\langle Q[Wh], K, P[A], Q[A] \rangle$ is "yes", with Q[A] the affirmative form of Q with A replacing the question word. Translating this CQI tuple in words, the answer to "Having retrieved information by querying K with Q[Wh], can a human reasonably infer Q[A] from P[A]?" is yes. In less formal words: after searching for an answer to a question in the text, does the most relevant passage justifies that A answers the question. This formulation captures the essence of factoid question answering, because both the retrieval phase and the answer validation phase are expressed.

CQI was intended for machine comprehension and specifically reading test comprehension, and is easily adapted for multiple-choice questions on short texts. The following is an example from CLEF 2014 Entrance exams test data. With K an informational text on monkeys (omitted here for space), Q is the sentence "It is difficult to study monkeys at play in cages because" and the shortened version "It is difficult to study monkeys at play in cages" is the hypothesis H. Then, each of the 4 answer choices is a potential text T, yielding 4 CQI instances:

1. it is hard to tell whether they are playing or not
2. they don't always play when scientists want them to
3. they play for such a long time that observers have to sit patiently
4. they prefer playing with humans to playing with other monkeys

Given how CQI is defined, there should be exactly one positive CQI instance if and only if there is exactly one correct answer to the question. CQI also encodes that the system has to read the text and find the passage relevant to the question, which is exactly what reading test comprehensions are all about.

Simple coreference resolution problems can also be encoded by CQI. If we have a first sentence $A[r_1, r_2]$ containing two potential referents r_1 and r_2, followed by a second sentence $B[p]$ containing a pronoun p. We can ask the two CQI problems $\langle \emptyset, A[r_1, r_2], B[p], B[r_1] \rangle$ and $\langle \emptyset, A[r_1, r_2], B[p], B[r_2] \rangle$. One should be positive and the other negative, corresponding to the correct and incorrect referents.

Winograd schemas (Levesque et al., 2011), a richer form of coreference resolution aiming at testing inference are naturally encoded by CQI. A typical Winograd schema is composed of a statement involving two parties, then an explanation introduced by "because" and referring with a pronoun to one of the two parties, and including a wording alternative enough to change the referent. For example:

(36) The city councilmen refused the demonstrators a permit because they [feared/advocated] violence. Who [feared/advocated] violence?

If "feared" is selected, the answer to the question is the councilmen. If "advocated", it is the demonstrators. There are several ways to turn this problem into CQI, the simplest is probably to answer one CQI instance with Q, K empty, "The city councilmen refused the demonstrators a permit" as H, and "the city councilmen advocated violence" as T and conclude. This example is a veiled textual entailment problem, but others do not contain "because" and can be seen as CQI following the above encoding for coreference resolution. This is an example of such a Winograd scheme:

(37) The sun was covered by a thick cloud all morning, but luckily, by the time the picnic started, it was [gone/out]. What was [gone/out]?

3.2 Proof system for CQI

The previous section presents the problem of Contextually Queried Inference (CQI). This section deals with solving this problem, first in a formal and theoretical setting, then from a more concrete standpoint with hints at an initial implementation.

3.2.1 The recursive nature of CQI

Contrary to RTE's, CQI's formulation already hints at how systems should proceed to deal with inference and provides them with natural guides: a context and

a query. These additional two elements capture a preliminary information gathering phase before trying to draw conclusions. In reading comprehension, a human reader exploring the text for the answer to a question will most likely need to draw other inferences and ask himself more questions. Likewise, when solving CQI, a system should be able to spawn from itself smaller and simpler CQI problems to either help the retrieval process or the inference resolution. This is not really feasible in RTE given the short length of the textual data. This is what we mean by the recursive nature of CQI: a large context and a query aiming at extracting a reduced portion of it precisely call for a recursive system.

In the next part, we present a formal proof system which theoretically solves CQI. The system works in a non-deterministic way and makes several assumptions which render its immediate concrete implementation unlikely at best. So why define such a theoretical system? It is very useful to reflect on the capabilities of real systems. We can abstract the rules we present from any tedious real-world considerations like algorithms, tools or resources. We can give this system as much freedom as we want in term of *chaining capabilities*: which rule can be applied after another. We can identify useful capabilities that might be completely ignored by current concrete implementations. Then we can express the inner workings of real systems in term of the rules of the formal system that they can implement and cannot, and in term of the intermediary processes they can spawn and solve and those they cannot even consider. In other words, this formal proof system is just another way of qualitatively assessing practical implementations dealing with inference problems.

3.2.2 Capabilities

The three capabilities presented in this section are the ones we deem essential, we do not pretend that they are universally essential, as that would be a very difficult statement to prove.

To solve CQI, a system must necessarily be able to *solve elementary inference steps*. Without going into detail about what is elementary inference and what is not, we will just assume that systems are able to reliably solve the simpler RTE instances. This assumption is actually close to the truth, given the high performance of a few complex systems participating to the RTE challenges (Table 2.2 p. 81). But this capability is obviously not enough if the CQI instance is harder than RTE, which it very well can be: for example, reading comprehension tests.

A second crucial capability is *recognizing a lack of a specific information*. This is what will drive the system to ask intermediary questions: if the system

detects that the information in the text or hypothesis is incomplete, it can create another query focusing on the specific lack and a new CQI problem is instantiated. The initial CQI problem will require this new one to be solved. The context is assumed complete as a simplification: there cannot be a lack of information when querying the context. Of course, in real problems, it sometimes happens that what we read is not enough to answer a question or draw a conclusion, but evaluations rarely focus on this issue and in our case, detecting a lack in the context could simply indicate to extend it and include more text or more data.

Finally, the system has to be able to *leverage external knowledge sources*. Indeed, we cannot expect all the required knowledge to be available in the context. Hence, it has access to what we call a background knowledge base (BKB), which we will simply represent by a collection of textual entailment rules of the form $T \longrightarrow H$, meaning T implies H. These general rules also include simple facts, by setting T to an empty text.

3.2.3 Proof system

The proof system we present in this section uses rules visually inspired from classic sequent calculus. There is a certain appeal to using a proof formalism in the context of textual inference: indeed, when solving a CQI problem, we should be constructing a step-by-step reasoning made out of other inferences and justified in the end by axioms representing elementary knowledge. The application of a single rule looks like this:

$$\frac{\text{Premise1} \qquad \text{Premise2}}{\text{Conclusion}}$$

It is both the rule and a single-step proof, and represents that whenever we have a proof of premise Premise1 and a proof of Premise2, then we have a proof of the Conclusion. It may be more natural to read the proof bottom-up to get an idea of how a system would work, the conclusion is the problem at hand, and the premises decompose that problem into simpler ones to solve.

Premises and conclusion are CQI instances $\langle Q, K, T, H \rangle$, under the syntax:

$$Q?K \vdash T \longrightarrow H$$

Without further ado, let us start with the axioms.

$$\frac{}{\emptyset?\emptyset \vdash T \longrightarrow H}\ ^{\text{te}}$$

The above rule is the basic textual entailment capability. When a system can ignore query and context, what remains is the resolution of a simple textual entailment problem, which is assumed to be done in one step.

$$\frac{}{Q?K \vdash A \longrightarrow B}\ \text{bkb}$$

The above rule is the ability for a system to use the inference rules in its background knowledge base. No matter the query or the context, if $A \longrightarrow B$ is a commonsense fact, true under most circumstances, it should be resolved in one step.

We then add two rules using directly the query or the context.

$$\frac{Q?K \vdash T \longrightarrow H}{Q?K \cup K' \vdash T \longrightarrow H}\ \text{ctxt-search}$$

$$\frac{\emptyset?K \vdash T \longrightarrow H}{Q?K \vdash T \longrightarrow H}\ \text{ctxt-found}$$

The first one enables the *search process*. The system can discard portions of the context (the K' in the conclusion) and continue querying on only the remaining context K. The second one discards the query, effectively marking the end of the search process. Remember that such a proof system is non-deterministic in nature. If a path exists to an elementary valid textual entailment (axiom), there is a combination of the above rules that will lead to it. If however the wrong choices are made, and we discard the query too soon, or discard the relevant passage, the textual entailment step at axiom level will generally fail, because it is not applied in the right context.

The next rule introduces variables.

$$\frac{Q'[X]?K, T \vdash P[a] \longrightarrow Q'[a] \qquad Q?K \vdash T[a] \longrightarrow H}{Q?K, X = a \vdash T[X] \longrightarrow H}\ \text{var-T}$$

Variable X represents a lack of information, in the text T in this case. The idea is to spawn another problem with a different query Q' about this variable, to find a plausible assignation of the variable in the final problem instance. The new subproblem's form will vary depending on the variable and the type of information itself, but we present it here as a simple generic question-answering problem on X. Eventually, an answer a is found, and the final context is modified to account for the assumption made on X. That way, the system keeps track of variables that

spawned sub-problems, and one can find the details of the sub-problem resolutions in the final result. The other premise assumes the variable is not present, which enables recursive resolution.

We can have a similar rule for H, or just consider $T \longrightarrow H$ as a whole. The following rule would replace the above rule and its symmetric "var-H" in a compact way.

$$\frac{Q'[X]?K, T, H \vdash P[a] \longrightarrow Q'[a] \qquad Q?K \vdash (T \longrightarrow H)[a]}{Q?K, X = a \vdash (T \longrightarrow H)[X]}\ \text{var-TH}$$

Finally, we need rules to enable the chaining of inferences.

$$\frac{Q?K \vdash A \longrightarrow C \qquad Q?K \vdash C \longrightarrow B}{Q?K \vdash A \longrightarrow B}\ \text{trans}$$

With the above rule, called "trans" for transitivity, we can solve a more complicated inference problem in several steps using the intermediary C. Note that with the "bkb" axiom (the second rule presented in this section), this rule enables the seamless integration of background knowledge to other inference processes.

3.2.4 Toward an implementation

The previous system remains very abstract and intentionally generic. It integrates all the capabilities we mentioned before, without constraining their implementation. There are several challenges to face when designing an implementation following this ruleset. This section presents these challenges, and also provides the opportunity to offer examples to illustrate how the proof system combines its rules.

3.2.4.1 Non-determinism

The proof system is intrinsically non-deterministic: there are a lot of different rules and sets of premises that lead to the same conclusion. So, starting from the CQI problem like a real system would, it would have to take multiple decisions over which rules to apply, in which order to explore the search space and which proof branches to prune.

Here are two examples of the same conclusion reached through different means.

$$\dfrac{\dfrac{}{Q?K \vdash T \longrightarrow A}\,\text{bkb} \qquad \dfrac{\dfrac{}{Q?K \vdash A \longrightarrow B}\,\text{bkb} \qquad \dfrac{}{Q?K \vdash B \longrightarrow H}\,\text{bkb}}{Q?K \vdash A \longrightarrow H}\,\text{trans}}{Q?K \vdash T \longrightarrow H}\,\text{trans}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset?\emptyset \vdash T \longrightarrow H}\,\text{te}}{\emptyset?K' \vdash T \longrightarrow H}\,\text{ctxt-search}}{Q?K' \vdash T \longrightarrow H}\,\text{ctxt-found}}{Q?K \vdash T \longrightarrow H}\,\text{ctxt-search}$$

The first system relies more on successive applications of background knowledge to build the inference, and the second goes through a more heavy retrieval process to end with the use of the basic textual entailment capability.

Potential solutions to the non-determinism include: smart heuristics to decide the order of rule applications, or parallel execution with all possible rules. We lean toward the second solution as the more future-proof approach. Classic systems operate pretty much like the second proof example: first perform a retrieval step, then apply a textual entailment module.

3.2.4.2 Robustness

As it stands, the application of a given rule is all-or-nothing: either it applies or it does not. This system is thus probably not very flexible. It could miss possible human-like inference steps because premises do not fit or there is no rule to exactly perform them.

Let us consider the following pseudo-rule:

$$\dfrac{A \longrightarrow C \qquad B \longrightarrow C}{A \longrightarrow B}$$

It is not included in our initial system because it is obviously incorrect in a logic sense. But in a linguistic and pragmatic one, there is a chance a link exists between A and B if they yield the same conclusion. “Ravaillac stabbed Henry IV to death” and “Ravaillac murdered Henry IV” both imply “Henry IV is dead”, and “Ravaillac stabbed Henry IV to death” implies “Ravaillac murdered Henry IV”. This rule is not to be applied everywhere nor to be trusted completely, but it can help in some cases. The previous system cannot handle this subtlety.

A solution to this problem would be to assign probabilities or at least weights to the rules, representing their confidence level given the confidence level of the

premises. This probabilistic framework would be harder to specify, but would probably be more flexible. Weights would also give insights on which rules to pursue first, which would help our first point.

3.2.4.3 Variable introduction

Finally, the variable introduction rule is hard to implement and is by far the less studied aspect of our system in current NLP. It is not easy to detect lack of information in the first place, and it is even harder to balance such a capability so that you only look for information that will help the system. Some features of current systems are comparable to this desirable capability, like coreference resolution. Other than that, potential solutions could be to look for missing arguments in frame semantics. For example, Framenet's lexical unit "hit" meaning "direct a blow at with one's hand or a tool or weapon" contains the "Instrument" and "Manner" arguments. When annotating a text and encountering "hit", if those arguments are absent, one could spawn CQI instances to find the instrument or the manner, because these elements could be useful in the inference problem.

3.3 Our contributions through the lens of CQI

The subsequent chapters detail our contributions to the implementation of various capabilities of the aforementioned system. We introduce them briefly in this section and evaluate the design of their inner workings relatively to what rules of the proof system they may indeed implement.

We remind the names of the rules in our proof system and a concise and concrete description of what their role is:

- `te`: solving the classic textual entailment problem, as posed by RTE.
- `bkb`: including a background knowledge base of any kind.
- `ctxt-search, ctxt-found`: retrieval of a useful fragment in the context, using the query.
- `var-T, var-H`: detecting a lack of information and spawning a subproblem.
- `trans`: chaining two inference steps.

Each contribution is designated by a short description, followed by the capabilities it is designed to implement in parenthesis. A fuller description is then provided, justifying why this set of capabilities is indeed implemented. We preface this introduction by admitting that none of our systems seeks lacks of information, and they are generally unable to spawn sub-problems equivalent in form to the initial task (no `var-T, var-H`).

A passage retrieval method using structured lexical expansion backed up by a dictionary (`ctxt-search, ctxt-found, bkb`)

We design a word similarity metric measuring what the dictionary definitions of two words have in common. The word similarity is easily extended to passage relevance rating with respect to a query, in order to select the best rated passage as the most relevant `ctxt-search, ctxt-found`. We do not use any arbitrary dictionary, but place this method in the context of a "simplification hypothesis": the simpler the lexicon, the easier it should be to compare the sense of two sentences. The dictionary must thus obey this simplification principle, so we pick the Simple English Wiktionary, a collaborative resource designed to use a simplified version of English. This is our lexical background knowledge base (`bkb`).

The method is evaluated on the Entrance Exams dataset at CLEF 2013, and on TREC QA Passages task. It has been published in a working note for CLEF 2013 (Gleize et al., 2013) and is detailed in this thesis in the first part of Chapter 4.

A textual entailment method using successive syntactical transformations extracted from a dictionary (`te, bkb, trans`)

The same simplification hypothesis is used as in the previous contribution, hence the same resource (Simple English Wiktionary). This approach transforms a Text into a Hypothesis, using paraphrase rules automatically extracted from the simplifying dictionary (`bkb`). Several paraphrase rules can be applied successively to further transform the initial sentence (`trans`). Features of the input textual entailment pairs are computed and used in a trained classifier to learned the textual entailment decision (`te`).

The method is evaluated on Semeval 2013 Task 7's dataset. It has been published in a working note at Semeval 2013 (Gleize and Grau, 2013) and is detailed in this thesis in the second part of Chapter 4.

A kernel method for recognizing sentence rewritings, with a notion of types able to encode lexical-semantic resources (`te, bkb`)

A kernel function on sentence pairs is defined: this means that it evaluates the similarity between two pairs of two sentences. It does so by computing the number of common rewriting rules within the first pair and the second pair. Each pair of sentences can be viewed as a textual entailment instance (`te`). Rewriting rules are really any 2 patterns of k-grams, and the kernel method actually learns which of those arbitrary rules is useful on the task and training data. Rewriting rules cannot directly be chained. To improve the performance, we allow rewriting rules to be equal modulo their *types*, which encode any kind of lexical-semantic variation the user chooses to add. In particular, one may add variations from background resources like WordNet (`bkb`). The kernel is then used in a SVM to produce a classifier.

The method is evaluated on paraphrase identification (MSR Paraphrase), textual entailment (RTE 3) (`te`), and answer sentence selection (a weak form of question answering) on a TREC dataset. It has been published at ACL 2015 (Gleize and Grau, 2015c) and at TALN 2015 (Gleize and Grau, 2015b). It is detailed in this thesis in the first part of Chapter 5.

A system for reading comprehension tests, with features extracted from a tree edit model and used in validation/invalidation classifiers (`ctxt-search, te, bkb, trans`)

This contribution describes the full system for automatically answering reading comprehension tests at the Entrance Exams task of CLEF 2015. We first use the question and answer choices as the query and find a ranking of relevant passages of the text. We are not too concerned about finding the correct passage at this point, as we believe this will be naturally done by the following processes (`ctxt-search`). Passages and answer choices are then viewed as their dependency trees, and a tree edit model is applied to try to turn the passage into the answer choice, using successive elementary tree edit operations (`trans`). Many such edits are possible at each step, so we use a beam search to reduce the search space. Features are extracted, including resources like WordNet and ConceptNet (`bkb`). Classifiers are trained using the features, on CLEF 2013-2014 data, to evaluate if a passage validates an answer choice, or if it invalidates. This last decision step can be seen as a textual entailment task (`te`). Our final answer to the multiple-choice question is then selected using the validation and invalidation scores of the choices.

This method is evaluated on the Entrance Exams task at CLEF 2015. It has been published in a working note at CLEF 2015 (Gleize and Grau, 2015a), with

some groundwork on validation and invalidation accomplished at CLEF 2014 (Gleize et al., 2014). It is detailed in this thesis in the last part of Chapter 5, as well as in Chapter 6.

Chapter 4

Structured lexical expansion

One needs several building blocks when trying to solve the problem of Contextually Queried Inference (CQI). Among them, a retrieval step uses the query to narrow down the useful portion of the context. Then, once the right context is found, a textual entailment step is used to solve what is left of the initial CQI instance. In this chapter, we describe our first approach to both mandatory steps of an ideal system. Having a common approach to solve the two phases, or at least two sharing many design similarities, tools and resources, is desirable when aiming for a later integration into a larger system. The methods can then share resources, pre-computed structures and algorithm outputs.

The following set of approaches actually predates the conception of CQI and in a way inspired it. In particular, it is in essence recursive and both methods can be applied indefinitely on sub-problems the same way they are applied to the initial problem. They try to modify the lexicon used in the sentences so that it is easier to handle and compare. For these reasons, the chapter is entitled "Structured Lexical Expansion". The first section presents our *simplification* hypothesis, then the second is dedicated to retrieval and finally the third to solving the textual entailment task.

4.1 The simplification hypothesis

Our hypothesis is that the simpler the language lexicon is, the easier it will be to access and compare meaning of sentences. This assumption is justified by the multiple attempts at controlled natural languages Schwitter (2010) and especially simplified forms of English. We expect both positive and negative effects from reducing the number of words in the lexicon. The major positive effect is that as the lexicon, and likely the grammar, are simplified, it is easier to model in a formal and exact way. The negative effect to overcome is the expressiveness of the language: some ideas will likely become harder to express, or will be expressed in lengthier sentences. It also does not get rid of ambiguity, and may in fact amplify this phenomenon as the more common words are even more employed than before and hence tasked with representing several of their possible senses more frequently. We however believe that this is a fair price to pay for a simpler surface language, and one that we will be able to handle when words are associated with each other within their context.

4.1.1 Simple English Wiktionary as a paraphrase resource

Simple English Wiktionary[1] is a collaborative dictionary written in a simplified form of English, primarily for children and English learners. It has adopted *Basic English* (Ogden, 1944) as its *de facto* language. Its definitions are clear, concise and get to the essence of the word without superfluous details. They seem fitted to acquire the "common sense knowledge" we need to solve open-domain inference problem.

The biggest flaw of this resource remains its low coverage: 24,330 words are defined at the time of writing. Also, its formatting presents inconsistencies which make it difficult to parse and use correctly, probably due to its collaborative nature. Guidelines for formatting definitions exist, but on one hand, they are not systematically applied and on the other hand, other untold conventions have surfaced. A Simple English Wiktionary parser thus has to be robust, and a "perfect" tool is likely too expensive to build relatively to the low coverage of the resource.

[1] `http://simple.wiktionary.org`

4.2 Querying the text

We first test our hypothesis on the task of passage retrieval of complex non-factoid question answering. Our main idea is that by simplifying the words of the query and the words of the text simultaneously, semantically relevant passages will surface more easily even when initially presenting to the inference system a distant surface form. We introduce a method based on recursive enriching of a word by the words of its definition in a dictionary with the property of simplification, like the Simple English Wiktionary (SEW). This method has been presented at CLEF 2013 in the Entrance Exams task (Gleize et al., 2013).

4.2.1 Dictionary-based passage retrieval

In a question answering system, passage retrieval aims at extracting from a relevant document the short text excerpt most likely to contain the answer. For the most realistic questions, direct matching of the surface form of the query and text sentences is not enough. As one of the most challenging and important processes in a QA system, passage retrieval would thus benefit from a more semantic approach.

We propose a passage retrieval method focusing on finding deep semantic links between words. The idea is that semantically-related words will share common words in their dictionary definitions. This is not a novel idea: Lesk (1986) defines a similarity measure based on the word overlap of WordNet gloses. But we go further and view a word as a kind of tree structure: the word itself is the root, and the words in its dictionary definition are the children. Recursively, we also use the dictionary definitions of these children to build a deeper tree.

Then, we use this word-tree structures to define a similarity measure. The idea is that semantically-related words will share descendants in their respective tree. And we can account for the descendants' depth to rate their "importance" in the similarity of the 2 initial words. Figure 4.1 (p. 99) shows an example. Both SEW definitions of the words "cat" and "wolf" contain the word "animal". But cat's definition contains "pet". It is not contained in the wolf's definition, but is however in the definition of one of its children: "dog". This is consistent with the idea that a wolf and a cat have more in common the fact of being animals, than the trait of being pets.

From this point forward we shall designate as *words* only lemmas from verbs, nouns, adjectives, adverbs and pronouns that are not stop-words. We assume a single purely textual document. *Document words* are words in the document. We

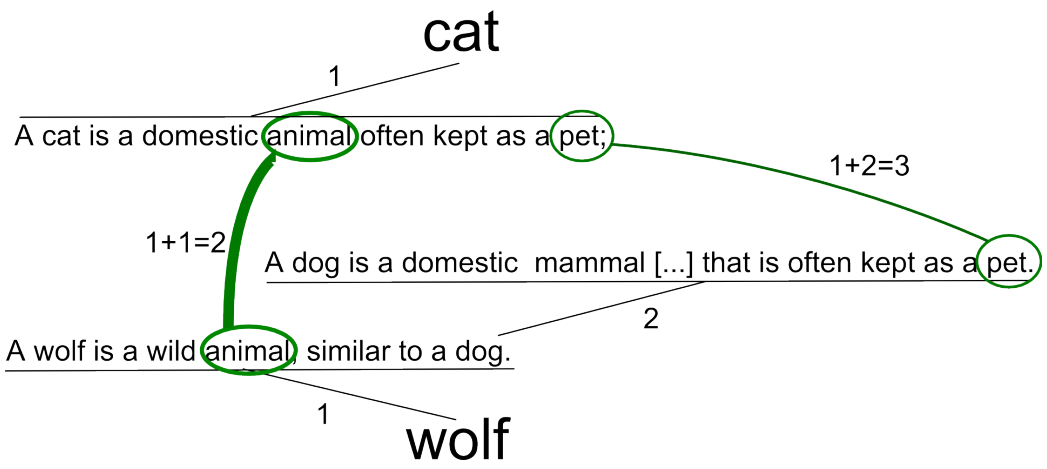

Figure 4.1: What have a cat and a wolf in common?

freely use the words children, descendants and ancestors to refer to the *word-tree* structure built out of dictionary definitions.

4.2.1.1 Pre-processing

We use Stanford CoreNLP (Manning et al., 2014) for POS-tagging and lemmatization. Penn Treebank tags are manually translated into their Wiktionary simplified counterparts (like "Noun", "Verb", "Adjective" or "Adverb"). This is all we need to access the Wiktionary entries of words, although they still may present different senses for the same part-of-speech.

4.2.1.2 Indexing the document

The document pre-processing phase builds a *semantic index* off of all the words in the document and their descendants in a given dictionary. This is similar to the index expansion of Attardi et al. (2012), except we use dictionaries and not background documents.

We look at all the document words consecutively. For each one, we build its word-tree up to a maximum depth of $d_{\max}$ (a parameter of the method). This is done just by looking up our dictionary and retrieving definitions. Then we build the *index*. The index contains an entry for each word w contained in a word-tree. So it can be a document word (because those are roots of trees) or a definition word found in the dictionary. The main information available for an entry is the

document word ancestors, $Anc(w)$. This is the list of document words whose word-tree contains w. For each document word in this list, we also record the depth of w in its tree. In other words, w is a descendant of all the words in $Anc(w)$.

Why do we record those document words? Because this list $Anc(w)$ is precisely a list of document words sharing a definition word (w) at some depth, which indicates that $Anc(w)$ are semantically related in some way. Back to our example from Figure 4.1, Anc("pet") would contain ("cat", 1) and ("wolf", 2). Anc("animal") would contain ("cat", 1) and ("wolf", 1). Just looking at those lists, we know that "cat" and "wolf" share some similarity.

Our indexing process is implemented so as to share a lot of its data structures to preserve computing time and space as much as possible. For example, the word-trees are implemented with pointers to index entries. That way, we only build as many nodes as words we encounter and index, and we do not build several for identical words.

This is the basis of our indexing, bar minor implementation details, like handling of multiple senses and POS-tags.

4.2.1.3 Passage retrieval

Our passage retrieval follows the following steps: we consider words of the question (or *query*), use the index to score their relevance, and combine these scores to rank candidate passages.

We build the *word-tree* for each word w_q in the query. For w in a tree, we have a corresponding entry in the index or not. If not, that means that w was not a definition word shared by any document word at any level of depth, so w is useless for finding matches for w_q in the document. If however, there is an entry in the index for w, it means that a document word shares this definition word with w_q. Not only one, but in fact, all the words in $Anc(w)$ share w with w_q. We can then compute a similarity *relatively to* w between w_q and those document words w_{doc_anc}, therefore rating the relevance of document words relatively to the query word:

$$Sim(w_q, w_{doc\_anc}, w) = idf(w_{doc\_anc}) \times \text{base}^{-(d_q + d_{doc\_anc})} \tag{4.1}$$

The depths d_q and d_{doc_anc} are the depths of w in the word-tree of respectively w_q and w_{doc_anc}. We choose $base$ depending on how strongly we want to penalize words as we go deeper in the tree. We found $base = 2$ to be a good start, but the final system uses a variable number based on children count: the intuition is

that the more words used in the definition of w, the less confident we are that each definition word is semantically related to w.

We compute the similarity for each w_q in the query and each w_{doc} in document word ancestors and sum over the w_q to obtain a relevance score for the document word:

$$Relevance(w_{doc}) = \sum_{w_q \in \text{ query}} \max_{w_{doc}, w} Sim(w_q, w_{doc}, w) \tag{4.2}$$

Finally, we select candidate passages with a sliding window of 3 consecutive sentences, we rank them by the sum of all the Relevances of their words normalized by word count. In our system, we also implemented a similar method to SiteQ's density-based scoring function (Lee et al., 2001): we add two consecutive Relevance scores of two words in the candidate passage and we normalize this by the square of their distance in the passage. This complexifies the indexing process quite a bit, but the details of the implementation would be too tedious to describe here.

4.2.2 Experiments

4.2.2.1 Data and evaluation methods

In our experiments, we work with two corpora of different nature. The first, and the one we are most interested in, is the set of reading comprehension tests provided by the task "Entrance exams" at CLEF (Peñas et al., 2013b) (cf Section 1.2.2). Short texts are provided with questions, and for each question, 4 answer choices are available, with exactly one correct. The length of the texts can vary between 60 sentences and 200 sentences. In general, questions are rather complex, and answer choices present an important lexical variation from the passages of the text they refer to, especially when the answer choice is the correct one. We evaluate our system for passage retrieval on the 9 reading tests (46 questions) of the test set of 2013, following Tellex's quantitative evaluation methodology (Tellex et al., 2003). We first annotate the test set in passages (which 2-to-4-sentence passage must be read to answer the question) to create a gold standard. The annotation with contiguous passages is quite straight forward, and only 2 questions need disjoint passages.

The second corpus is from the Question Answering track of TREC, specifically its Passages Task at TREC 2003. For each question, the top 1000 documents in the ACQUAINT corpus are provided, and the gold standard is a short sentence

from one of the documents. Questions are of factoid nature and are quite simple, but the pool of passage candidates is important in size, so the task remains challenging.

All the experiments report mainly three types of evaluation measures:

- MRR: mean reciprocal rank
- p@n: number of correct passages found in the top n
- nf: number of correct passages which were not found at all

4.2.2.2 Results

We are most interested in the results on the Entrance exams dataset: given the difficulty of the questions and the lack of background knowledge, passage retrieval quickly appear as a strong bottleneck for any question-answering system attempting to solve this task.

We implement several runs:

- MITRE as a weak baseline: simple word overlap algorithm (Light et al., 2001)
- SiteQ as a strong baseline: sentences are weighted based on query term density (Lee et al., 2001)
- SI(d_{max}), our Simple English Wiktionary-based indexing system, parameterized by d_{max}

We experiment on three types of queries:

- Question alone: this only uses the question.
- Question + Correct answer: the question and the correct answer are concatenated to form one query.
- Question + Incorrect answer: the question and each incorrect answer are concatenated to form several queries.

Results for Entrance Exams are shown on Table 4.1. Our system outperforms both baselines significantly on all types of queries (question alone, question + right answer, question + wrong answer) and measures. The difference is most noticeable when the systems do not have access to choices of answers, which is really

what we seek for the broader view of question answering. The performances is globally higher when the system uses the correct answer instead of wrong answers. This is not as helpful as it seems: on the real task, we do not know which is the correct answer and which are the incorrect ones. This observation only means that the words of the correct answer choices will be more likely to lead to the right passage than the incorrect answer choices, which is, at best, reassuring. What is also interesting is the increase in performance for SI as we increase the maximum depth of search in the dictionary. This seems to confirm that Simple English Wiktionary fits this task well and that our scoring functions scale correctly with the amount of knowledge that it provides.

Algorithm	MRR	p@1	p@3	p@5	p@10	nf
Question alone						
MITRE	0.215	0.13	0.20	0.26	0.37	**0.13**
SiteQ	0.337	0.22	0.39	0.52	0.61	0.37
SI(1)	0.355	0.22	0.43	0.59	0.69	0.28
SI(2)	0.392	0.24	**0.46**	**0.63**	**0.76**	0.20
SI(3)	**0.420**	**0.28**	**0.46**	**0.63**	0.74	**0.20**
Question + Correct answer						
MITRE	0.320	0.20	0.33	0.49	0.50	**0**
SiteQ	0.506	**0.37**	0.57	0.65	0.89	0.07
SI(3)	**0.523**	0.35	**0.65**	**0.74**	**0.93**	**0.07**
Question + Incorrect answer						
MITRE	0.254	0.14	0.22	0.31	0.53	**0**
SiteQ	0.466	0.32	0.54	0.62	0.81	0.09
SI(3)	**0.480**	**0.33**	**0.57**	**0.72**	**0.83**	0.15

Table 4.1: Evaluation of passage retrieval on QA4MRE 2013 Entrance exam task

Our results on TREC are shown on Table 4.2. On this dataset, we can compare with existing passage retrieval methods, like the one presented by Cui et al. (2005). They report scores for the baselines (MITRE, SiteQ and NUS). Their system is described in our literature review (page 62), but we remind the reader here that it implements matching of dependency relation paths, with a learning method to fuzzily align dependencies. They report results for strict matching (Rel_strict in our table), without the learned alignments, and for expectation maximization (Rel_EM in our table). Their system actually re-rank passages already processed by the baselines MITRE and NUS, hence the indication between parenthesis. For

a fairer comparison with (Cui et al., 2005), we add dependency bigrams in our scoring model. We call dependency bigrams all the pairs in the sentence of words linked by a dependency relation. For a pairs of document words (w_1, w_2), we use the formula described in Equation 4.2 and the distance in position of the 2 words (dist) to compute the relevance of the dependency bigram:

$$Relevance(w_1, w_2) = \frac{Relevance(w_1) * Relevance(w_2)}{dist(w_1, w_2)} \tag{4.3}$$

We report the results of our Semantic Indexing method with Simple English Wiktionary as the dictionary, with a depth varying from 0 (no dictionary words) to 3, without or with dependency bigrams. We first see that our method vastly outperforms all the baselines. Adding levels of depth seems to scale correctly from 0 to 2, but at depth 3 we start to notice an MRR and p@1 loss. Adding dependency bigrams improve the results a lot which proves that adding structure, even in the most basic way, can help passage retrieval. It also helps us match the strict relation matching method of Cui et al., but with training, their fuzzy relation matching gets far ahead of our results. Using trained parameters in our scoring functions could probably have helped but we also think that experiments on TREC do not benefit as much from lexical expansion (which Cui et al. (2005) does not implement). Our method was designed primarily for the Entrance Exams task, and we use it in the evaluation campaigns of CLEF 2013 and CLEF 2014.

4.2.2.3 Conclusion

In this part, we designed and implemented a passage retrieval method based on recursive lexical expansion using freely available dictionaries. Our scoring function shows some promise on two evaluation datasets, and especially seems to scale correctly with how deep we look up in the dictionary. Its limitations are visible on the second evaluation task (TREC), where a method from 2005 outperforms it by using dependency relation matching and learning.

If we want to refer to our inference hierarchy (Section 1.3.2 p. 29), our previous system stayed at the Tier 1 of methods, and did not use the structure of the sentence, although it did use a structured view of the lexicon. The only attempt at integrating sentence structure was adding dependency relation bigrams, and it improved our results a lot. In the next section, we keep the same simplification hypothesis and resources but go up a Tier in the hierarchy. We also switch to another elementary task described in contextually queried inference: solving the

Algorithm	MRR	p@1
Baseline		
MITRE	0.2	0.1235
SiteQ	0.2765	0.1975
NUS	0.2677	0.1759
(Cui et al., 2005)		
Rel_strict (MITRE)	0.299	0.2253
Rel_strict (NUS)	0.3625	**0.2716**
Rel_EM (MITRE)	0.4218	0.3457
Rel_EM (NUS)	**0.4761**	**0.3889**
SI, no bigrams		
SI(0)	0.3035	0.1927
SI(1)	0.3244	0.2141
SI(2)	0.3331	0.211
SI(3)	0.3298	0.211
SI, with bigrams		
SIb(0)	0.3208	0.1988
SIb(1)	0.3511	0.2171
SIb(2)	**0.3640**	0.2355
SIb(3)	0.3583	0.2324

Table 4.2: Evaluation of passage retrieval on TREC 2003, Passages Task

textual entailment step, which will benefit more from taking into account sentence structure than the task of passage retrieval.

4.3 Solving the inference step

In this section, we describe a textual entailment method, based on substitution by Basic English variants. Basic English paraphrases are acquired from the Simple English Wiktionary. Substitutions are applied both on the text and the hypothesis in order to reduce the diversity of their vocabulary and map them to a common vocabulary. The evaluation of our approach on the SemEval 2013 Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge data shows promising results, and this work is a first step toward an open-domain system able to exhibit composable text understanding capabilities as imagined in Chapter 3. This work has been published at SemEval 2013 (Gleize and Grau, 2013).

4.3.1 Introduction

Automatically assessing student answers is a challenging natural language processing task (NLP). It is a way to make test grading easier and improve adaptive tutoring (Dzikovska et al., 2010), and is the goal of the SemEval 2013's task 7, titled *Joint Student Response Analysis*. More specifically, given a question, a known correct "reference answer" and a 1- or 2-sentence student answer, the goal is to determine the student's answer accuracy (Dzikovska et al., 2013). This can be seen as a paraphrase identification problem between student answers and reference answers.
Paraphrase identification searches whether two sentences have essentially the same meaning (Culicover, 1968). Automatically generating or extracting semantic equivalences for the various units of language – words, phrases, and sentences – is an important problem in NLP and is being increasingly employed to improve the performance of several NLP applications (Madnani and Dorr, 2010), like question-answering and machine translation.
Paraphrase identification would benefit from a precise and broad-coverage semantic language model. This is unfortunately difficult to obtain to its full extent for any natural language, due to the size of a typical lexicon and the complexity of grammatical constructions. This is why we choose to test our simplification hypothesis on this task as well.

Our method starts with acquiring paraphrases from the Simple English Wiktionary's definitions. Using those, we generate variants of both sentences whose meanings are to be compared. Finally, we compute traditional lexical and semantic similarity measures on those two sets of variants to produce features to train a classifier on the SemEval 2013 datasets in order to take the final decision.

4.3.2 Acquiring simplifying paraphrases

Simple Wiktionary word definitions are different from usual dictionary definitions. Aside from the simplified language, they often prefer to give a complete sentence where the word – e.g. a verb – is used in context, along with an explanation of what it means. To define the verb *link*, Simple Wiktionary states that *If you link two or more things, you make a connection between them*, whereas the standard Wiktionary uses the shorter and more cryptic *To connect two or more things*.

We notice in this example that the definition from Simple Wiktionary consists of two clauses, linked by a subordination relation. It is actually the case for a lot of

Word (POS-tag)	Word part	Defining part
link (V)	you link two or more things	you make a connection between them
giraffe (N)	the giraffe	the tallest land animal in the world
bright (Adj)	something is bright	it gives out or fills with much light

Table 4.3: Word part and defining part of some Simple Wiktionary definitions

verb definitions: a quick statistical study shows that 70% of these definitions are composed of two clauses, an independent clause, and a subordinate clause (often an adverbial clause). One clause illustrates how the verb is used, the other gives the explanation and the actual dictionary definition, as in the previous example. Using this structural pattern did not make much sense with our previous semantic indexing method for passage retrieval, but the structure of these dictionary definitions are at the basis of our method for acquiring paraphrases.

4.3.2.1 Pre-processing

We use the Stanford Parser to parse the definitions and get a dependency graph (De Marneffe and Manning, 2008). Using a few hand-written rules, we then retrieve both parts of the definition, which we call the *word part* and the *defining part* (see table 4.3 page 107 for examples). We can do this for definitions of verbs, but also for nouns, like *the giraffe is the tallest land animal in the world* to define *giraffe*, or adjectives, like *if something is bright it gives out or fills with much light* to define *bright*. We only provide the details of our method for processing verb definitions, as they correspond to the most complex cases, but we proceed similarly for noun, adjective and adverb definitions.

4.3.2.2 Argument matching

Word and defining parts alone are not paraphrases, but we can obtain phrasal paraphrases from them. If we see word part and defining part as two semantically equivalent predications, we have to identify the two predicates with their arguments, then match arguments with corresponding meaning, i.e. match arguments which designate the same entity or assume the same semantic function in both parts, as showed in Table 4.4.

For verb definitions, we identify the predicates as the main verbs in both clauses (hence *link* matching with *make* in table 4.4) and their arguments as a

you	$\rightarrow$	you
link	$\rightarrow$	make
$\emptyset$	$\rightarrow$	a connection
$\emptyset$	$\rightarrow$	between
two or more things	$\rightarrow$	them

Table 4.4: Complete matching for the definition of verb *link*

POS-filtered list of their syntactic descendants. Then, our assumption is that every argument of the word part predicate is present in the defining part, and the defining part predicate can have extra arguments (like *a connection*).

We define $s(A, B)$, the *score* of the pair of arguments (A, B), with argument A in the word part and argument B in the defining part. We then define a *matching* M as a set of such pairs, such that every element of every possible pair of arguments is found at most one time in M. A *complete matching* is a matching M that matches every argument in the word part, i.e., for each word part argument A, there exists a pair of arguments in M which contains A. Finally, we compute the *matching score* of M, $S(M)$, as the sum of scores of all pairs of M.

The *score* function $s(A, B)$ is a linear combination of the following features computed on a pair of arguments (A, B):

- Raw string similarity. Sometimes the same word is reused in the defining part or words share the same stem.
- Having an equal/compatible dependency relation with their respective main verb. For example, we consider direct and indirect object as compatible dependencies, but not the same.
- Relative position in clause. The relative position of a word is its position divided by the length of the sentence.
- Depth difference in parsing tree. These last 3 features assess if the two arguments play the same syntactic role.
- Same gender and number. If different, it is unlikely that the two arguments designate the same entity.
- If (A, B) is a pair (noun phrase, pronoun). We hope to capture an anaphoric expression and its antecedent.

Feature	Weight
Exact word (b)	100
Same stem (b)	30
Same dependency with verb (b)	20
Compatible dependency (b)	5
Relative position difference	5
Depth difference in parse tree	1
Same number (plural)	20
Same number (singular)	10
Same gender (if not neutral) (b)	30
Same gender (if neutral) (b)	5
Noun/pronoun pair (b)	10
WordNet similarity	30

Table 4.5: Feature weights

- WordNet similarity by Lin (1998). (cf Section 2.2.2.2 p. 54). If words belong to close synsets, they're more likely to identify the same entity.

The weights in the linear sum are set following the Table 4.5. "(b)" indicates a boolean feature, which can be equal to 1 or 0.

4.3.2.3 Phrasal paraphrases

We compute the complete matching M which maximizes the matching score $S(M)$. Although it is possible to enumerate all matchings, it is intractable; therefore when predicates have more than 4 arguments, we prefer constructing a best matching with a beam search algorithm. After replacing each pair of arguments with linked variables, and attaching unmatched arguments to the predicates, we finally obtain phrasal paraphrases of this form:

$$\langle\ X \text{ link } Y, X \text{ make a connection between } Y\ \rangle$$

We extract about 20,650 paraphrases from the Simple Wiktionary using this method.

4.3.3 Paraphrasing exercise answers

4.3.3.1 Paraphrase generation and pre-ranking

Given a sentence, and our Simple Wiktionary paraphrases, we can generate sentential paraphrases by simple syntactic pattern matching –and do so recursively by taking previous outputs as input–, with the intent that these new sentences use increasingly more Basic English. Figure 4.2 illustrates the whole process. We generate as many variants starting from both reference answers and student answers as we can in a fixed amount of time, as an anytime algorithm would do. We prioritize substituting verbs and adjectives over nouns, and non Basic English words over Basic English words. Given a student answer and reference answers, we then use a simple Jaccard distance (on lowercased lemmatized non-stopwords) to score the closeness of student answer variants to reference answer variants: we measure how close the vocabulary used in the two statements has become. For each reference answer A, we keep only the n closest variants of the student answer to A's variant set (in our experiments, $n = 10$). We finally rank the reference answers according to the average distance from their n closest variants to A's variant set and keep the top-ranked one for our classification experiment.

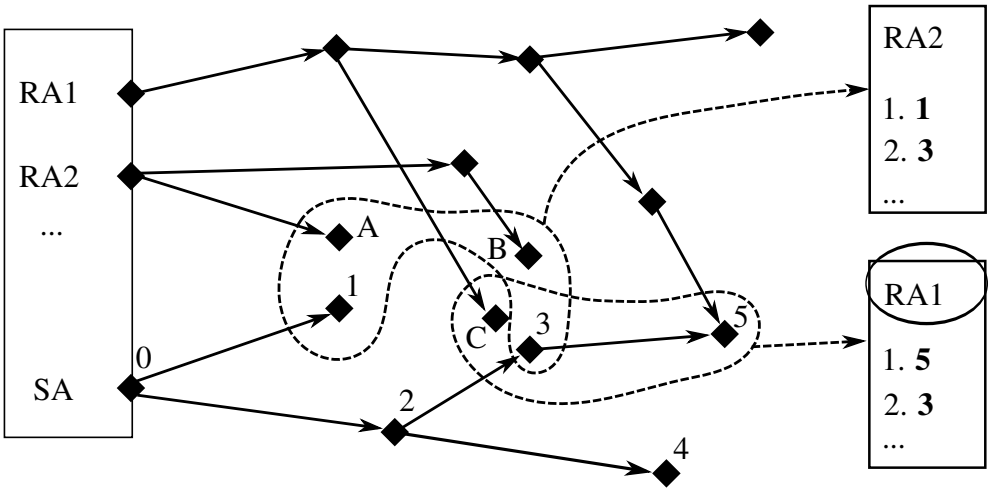

Figure 4.2: Variants are generated from all reference answers (RA) and the student answer (SA). For each reference answer RA, student answer variants are ranked based on their lexical distance from the variants of RA. The reference with the n closer variants to the student variants is kept (here: RA1).

4.3.3.2 Classifying student answers

SemEval 2013 task 7 offers 3 problems: a 5-way task, with 5 different answer judgements, and 3-way and 2-way tasks, conflating more judgement categories each time. Two different corpora, Beetle and SciEntsBank, were labeled with the 5 following labels: Correct, Partially_correct_incomplete, Contradictory, Irrelevant and Non_Domain, as described in Dzikovska et al. (2012). We see the n-way task as a n-way classification problem. The instances of this problem are the pairs (student answer, reference answer).

We compute for each instance the following features: For each of the n closest variants of the student answer to some variant of the reference answer computed in the pre-ranking phase:

- Jaccard similarity coefficient on non-stopwords.
- A boolean representing if the two statements have the same polarity or not, where polarity is defined as the number of *neg* dependencies in the Stanford Parser dependency graph.
- Number of "paraphrasing steps" necessary to obtain the variant from a raw student answer.
- Highest WordNet similarity of their respective nouns.
- WordNet similarity of the main verbs.

General features:

- Answer count (how many students typed this answer), provided in the datasets.
- Length ratio between the student answer and the closest reference answer.
- Number of (non-stop)words which appear neither in the question nor the reference answers.

We train an SVM classifier (with a one-against-one approach to multiclass classification) on both Beetle and SciEntsBank, for each n-way task.

System	Beetle unseen answers	SciEntsBank unseen questions
Majority	0.4010	0.4110
Lexical overlap	0.5190	0.4130
Mean	0.5326	0.4078
ETS-run-1	0.5740	**0.5320**
ETS-run-2	**0.7150**	0.4010
Simple Wiktio	0.5330	**0.4820**

Table 4.6: SemEval 2013 evaluation results.

4.3.3.3 Evaluation

Table 4.6 presents our system's overall accuracy on the 5-way task, along with the top scores at SemEval 2013, mean scores, and baselines –majority class and lexical overlap– described in Dzikovska et al. (2012).

Our system performs slightly better in overall accuracy on Beetle unseen answers and SciEntsBank unseen questions than both baselines and the mean scores. While results are clearly below the best system trained on the Beetle corpus questions, we hold the third best score for the 5-way task on SciEntsBank unseen questions, while not fine-tuning our system specifically for this corpus. This is rather encouraging as to how suitable Simple Wiktionary is as a resource to extract open-domain knowledge from.

4.3.4 Discussion

The system we present in this section is a step towards an open-domain machine reading system capable of understanding and reasoning as envisioned in Chapter 3.

Direct modeling of the semantics of a full natural language appears too difficult. We therefore decide to first project the English language onto a simpler English, so that it is easier to model and draw inferences from. The projection is done by applying several paraphrase rules consecutively. In a way, this system proposes an instanciation of two rules proposed in Chapter 3, the textual entailment axiom and the inference chaining, reminded to the reader below:

$$\frac{}{\emptyset ? \emptyset \vdash T \longrightarrow H}\ \text{te}$$

$$\frac{Q ? K \vdash A \longrightarrow C \qquad Q ? K \vdash C \longrightarrow B}{Q ? K \vdash A \longrightarrow B}\ \text{trans}$$

In this contribution, we do not take into account that texts present information lacks: missing information that cannot be inferred by reasoning on the text alone, but requires a certain amount of background knowledge. On open-domain, this is quite hard to detect and to our knowledge has never been attempted. On closed-domain however, Peñas and Hovy (2010) show that these gaps can be filled by maintaining a background knowledge base built from a large corpus.

Although Simple Wiktionary is not a large corpus by any means, it can serve our purpose of acquiring basic knowledge for assessing exercise answers, which present similarities to closed-domain data, and has the advantage to be in constant evolution and expansion, as well as interfacing very easily with the richer Wiktionary and Wikipedia.

4.4 Conclusion

This chapter dealt with a method of structured lexical expansion, first applied at word level on passage retrieval, then at sentence level on textual entailment. While performing rather well, it requires a lot of fine-tuning to obtain performance comparable to early retrieval methods – on passage retrieval – and methods using only surface forms – on textual entailment. The resource backing the method has a rather poor coverage relatively to today's standards, which naturally translates into a poor robustness.

One of the weaknesses of previous systems is that the way we perform the transformations on the sentence, either lexical or structural, is determined by hand. It means that the methods will not be able to translate easily on other datasets, and will at some point fail on words or patterns ignored by our rules: for example, when the word is not included in the dictionary, or when the correct paraphrase is not contained in our paraphrase bank.

The idea of a structured and iterative transformation of a word or sentence remains a very appealing point, certainly in line with what Chapter 3 deems as essential. Indeed, viewing the task of detecting textual inference as finding a valid rewriting sequence between two sentences is natural. The next chapter extends on

this idea to present machine learning methods for specifically rewriting sentences, applied to CQI-related tasks. These methods retain the ability to include lexical-semantic resources, as used in this chapter, but can also be trained on standard datasets to learn which rule to use when turning a sentence into another.

Chapter 5

Sentence rewriting as a machine learning task

In this chapter, we explore in a deeper way the concept of *sentence rewriting*, the way a sentence can be transformed into another. We are of course not interested in any kind of transformation, but only those which preserve some of the sense in the initial sentence: ideally, we only want transformations such that the initial sentence entails the final sentence. This chapter does not address the integration of the query and the context defined in Chapter 3 and focuses solely on the textual entailment step, with possibly some form of chaining of inference rules.

We build on the ideas of the previous chapter, especially the applications of paraphrase rules to rewrite sentences. While our previous systems were not really robust and were too much tied to one resource and to specific syntactical patterns, in the following we design and implement two methods with several new advantages. They do not require much fine-tuning, they allow the easy integration of a variety of lexical and semantic resources, and can be used on multiple CQI-related tasks and datasets. The general idea is to define a vast number of potential rewriting rules, each with interesting properties and potentially including lexical-semantic resources, but whose application may introduce noise in the inference. Which rules are used and how they are applied is then learned on standard training data.

The first method is based on string kernels and performs well on three tasks (RTE, answer sentence selection and paraphrase identification). As a kernel method, it does not scale well to huge training datasets. And as a string-based method, it actually does require a decent amount of training data to be effective.

This duality is addressed by the second method. It is based on a tree-edit

model and obtains the second best performance at CLEF 2015's Entrance exams. Like the string kernel method, it also deals with sentence rewriting, as tree edit operations indeed model some form of sentence transformations. But in particular, it is designed to not require much training data to be effective, and instead relies on efficient tree heuristics and a comprehensive set of edit features.

5.1 A Unified Kernel Approach for Learning Typed Sentence Rewritings

The inference problems we are interested in can be framed as determining if two given sentences are a rewriting of each other. In this section, we propose a class of kernel functions, referred to as type-enriched string rewriting kernels, which, used in kernel-based machine learning algorithms, allow to learn sentence rewritings. Unlike previous work, this method can be fed external lexical semantic relations to capture a wider class of rewriting rules. It also does not assume preliminary syntactic parsing but is still able to provide a unified framework to capture syntactic structure and alignments between the two sentences. We experiment on three different natural sentence rewriting tasks and obtain state-of-the-art results for all of them. This work has been published at ACL 2015 (Gleize and Grau, 2015c) and TALN 2015 (Gleize and Grau, 2015b).

5.1.1 Introduction

Detecting implications of sense between statements can naturally be framed as classification tasks, and as such most current solutions make use of supervised machine learning. They have to tackle several challenges: picking an adequate language representation, aligning semantically equivalent elements and extracting relevant features to learn the final decision. Bag-of-words and by extension bag-of-ngrams are traditionally the most direct approach and features rely mostly on lexical matching (Wan et al., 2006; Lintean and Rus, 2011; Jimenez et al., 2013). Moreover, a good solving method has to account for typically scarce labeled training data, by enriching its model with lexical semantic resources like WordNet (Miller, 1995) to bridge gaps between surface forms (Mihalcea et al., 2006; Islam and Inkpen, 2009; Yih et al., 2013). Models based on syntactic trees remain the typical choice to account for the structure of the sentences (Heilman and Smith, 2010; Wang and Manning, 2010; Socher et al., 2011; Calvo et al., 2014). Usually the best systems manage to combine effectively different methods, like Madnani et al.'s meta-classifier with machine translation metrics (Madnani et al., 2012).

A few methods (Zanzotto et al., 2007, 2010; Bu et al., 2012) use kernel functions to learn what makes two sentence pairs similar. Building on this work, we present a type-enriched string rewriting kernel giving the opportunity to specify in a fine-grained way how words match each other. Unlike previous work, rewrit-

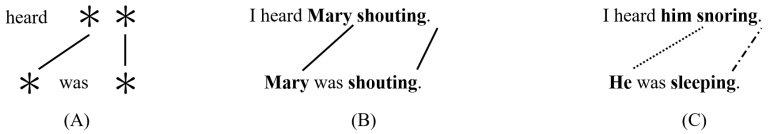

Figure 5.1: Rewriting rule (A) matches pair of strings (B) but does not match (C).

ing rules learned using our framework account for local syntactic structure, term alignments and lexical-semantic typed variations in a unified approach.

Going back to our formal CQI system in Chapter 3, this method does not directly allow the generalized chaining of rewriting rules:

$$\frac{A \longrightarrow C \qquad C \longrightarrow B}{A \longrightarrow B}$$

What is does model very well however is local transformations inside the same sentence, with varying spanning length:

$$\frac{A_1 \longrightarrow B_1 \qquad A_2 \longrightarrow B_2}{A_1 A_2 \longrightarrow B_1 B_2}$$

The next two sections describe the definition of the kernel and its efficient implementation.

5.1.2 Type-Enriched String Rewriting Kernel

Kernel functions measure the similarity between two elements. Used in machine learning methods like SVM, they allow complex decision functions to be learned in classification tasks (Vapnik, 2000). The goal of a well-designed kernel function is to have a high value when computed on two instances of same label, and a low value for two instances of different label.

5.1.2.1 String rewriting kernel

String rewriting kernels (Bu et al., 2012) count the number of common rewritings between two pairs of sentences seen as sequences of words. The rewriting rule (A) in Figure 5.1 can be viewed as a kind of phrasal paraphrase with linked variables (Madnani and Dorr, 2010). Rule (A) rewrites (B)'s first sentence into its second but it does not however rewrite the sentences in (C), which is what we try

to fix in this work.
Following the terminology of string kernels, we use the term *string* and *character* instead of *sentence* and *word*. We denote $(s,t) \in (\Sigma^* \times \Sigma^*)$ an instance of string rewriting, with a source string s and a target string t, both finite sequences of elements in Σ the finite set of characters. Suppose that we are given training data of such instances labeled in $\{+1,-1\}$, for paraphrase/non-paraphrase or entailment/non-entailment in applications. We can use a kernel method to train on this data and learn to automatically classify unlabeled instances. A kernel on string rewriting instances is a map:

$$K : (\Sigma^* \times \Sigma^*) \times (\Sigma^* \times \Sigma^*) \to \mathbb{R}$$

such that for all $(s_1,t_1),(s_2,t_2) \in \Sigma^* \times \Sigma^*$,

$$K((s_1,t_1),(s_2,t_2)) = \langle \Phi(s_1,t_1), \Phi(s_2,t_2) \rangle \tag{5.1}$$

where Φ maps each instance into a high dimension feature space. Kernels allow us to avoid the potentially expensive explicit representation of Φ through the inner product space they define. The purpose of the string rewriting kernels is to measure the similarity between two pairs of strings in term of the number of rewriting rules of a set R that they share. Φ is thus naturally defined by $\Phi(s,t) = (\phi_r(s,t))_{r \in R}$ with $\phi_r(s,t) = n$ the number of contiguous substring pairs of (s,t) that rewriting rule r matches.

5.1.2.2 Typed rewriting rules

Let the wildcard domain $D \subseteq \Sigma^*$ be the set of strings which can be replaced by wildcards. We now present the formal framework of the type-enriched string rewriting kernels.
Let Γ_p be the set of *pattern types* and Γ_v the set of *variable types*.
To a type $\gamma_p \in \Gamma_p$, we associate the *typing relation* $\overset{\gamma_p}{\approx} \subseteq \Sigma \times \Sigma$.
To a type $\gamma_v \in \Gamma_v$,we associate the *typing relation* $\overset{\gamma_v}{\rightsquigarrow} \subseteq D \times D$.
Together with the typing relations, we call the association of Γ_p and Γ_v the *typing scheme* of the kernel. Let Σ_p be defined as

$$\Sigma_p = \bigcup_{\gamma \in \Gamma} \{[a|b] \mid \exists a,b \in \Sigma, a \overset{\gamma}{\approx} b\} \tag{5.2}$$

We finally define typed rewriting rules. A *typed rewriting rule* is a triple $r = (\beta_s, \beta_t, \tau)$, where $\beta_s, \beta_t \in (\Sigma_p \cup \{*\})^*$ denote source and target string typed patterns and $\tau \subseteq ind_*(\beta_s) \times ind_*(\beta_t)$ denotes the alignments between the wildcards

in the two string patterns. Here $ind_*(\beta)$ denotes the set of indices of wildcards in β.
We say that a rewriting rule (β_s, β_t, τ) *matches* a pair of strings (s, t) if and only if the following conditions are true:

- string patterns β_s, resp. β_t, can be turned into s, resp. t, by:
  - substituting each element $[a|b]$ of Σ_p in the string pattern with an a or b ($\in \Sigma$)
  - substituting each wildcard in the string pattern with an element of the wildcard domain D
- $\forall (i, j) \in \tau$, s, resp. t, substitutes the wildcards at index i, resp. j, by $s_* \in D$, resp. t_*, such that there exists a variable type $\gamma \in \Gamma_v$ with $s_* \overset{\gamma}{\rightsquigarrow} t_*$.

A *type-enriched string rewriting kernel* (TESRK) is simply a string rewriting kernel as defined in Equation 5.1 but with R a set of typed rewriting rules. This class of kernels depends on wildcard domain D and the typed rewriting rules R which can be tuned to allow for more flexibility in the matching of pairs of characters in a rewriting rule.
Within this framework, the k-gram bijective string rewriting kernel (kb-SRK) is defined by the wildcard domain $D = \Sigma$ and the ruleset

$$R = \{(\beta_s, \beta_t, \tau) \mid \beta_s, \beta_t \in (\Sigma_p \cup \{*\})^k, \tau \text{ bijective}\}$$

under $\Gamma_p = \Gamma_v = \{id\}$ with $a \overset{id}{\approx} b$, resp. $a \overset{id}{\rightsquigarrow} b$, if and only if $a = b$.

We now present an example of how kb-SRK is applied to real pairs of sentences, what its limitations are and how we can deal with them by reworking its typing scheme. Let us consider again Figure 5.1, (A) is a rewriting rule with $\beta_s =$ (*heard*, $*$, $*$), $\beta_t =$ ($*$, *was*, $*$), $\tau = \{(2, 1); (3, 3)\}$. Each string pattern has the same length, and pairs of wildcards in the two patterns are aligned bijectively. This is a valid rule for kb-SRK. It matches the pair of strings (B): each aligned pair of wildcards is substituted in source and target sentences by the same word and string patterns of (A) can indeed be turned into pairs of substrings of the sentences. However, it cannot match the pair of sentences (C) in the original kb-SRK. We change Γ_p to {hypernym, id} where $a \overset{hypernym}{\approx} b$ if and only if a and b have a common hypernym in WordNet. And we change Γ_v to $\Gamma_v =$ {same_pronoun, entailment, id} where $a \overset{same\_pronoun}{\rightsquigarrow} b$ if and only if a and b are a pronoun of the

same person and same number, and $a \overset{entailment}{\rightsquigarrow} b$ if and only if verb a has a relation of entailment with b in WordNet.
By redefining the typing scheme, rule (A) can now match (C).

5.1.3 Computing TESRK

5.1.3.1 Formulation

The k-gram bijective string rewriting kernel can be computed efficiently (Bu et al., 2012). We show that we can compute its type-enriched equivalent at the price of a seemingly insurmountable loosening of theoretical complexity boundaries. Experiments however show that its computing time is of the same order as the original kernel.
A type-enriched kb-SRK is parameterized by k the length of k-grams, and its typing scheme the sets Γ_p and Γ_v and their associated relations. The annotations of Γ_p and Γ_v to K_k and $\bar{K}_k$ will be omitted for clarity and because they typically will not change while we test different values for k.
We rewrite the inner product in Equation 5.1 to better fit the k-gram framework:

$$K_k((s_1,t_1),(s_2,t_2)) = \sum_{\substack{\alpha_{s_1}\in\text{k-grams}(s_1)\\ \alpha_{t_1}\in\text{k-grams}(t_1)}} \sum_{\substack{\alpha_{s_2}\in\text{k-grams}(s_2)\\ \alpha_{t_2}\in\text{k-grams}(t_2)}} \bar{K}_k((\alpha_{s_1},\alpha_{t_1}),(\alpha_{s_2},\alpha_{t_2})) \tag{5.3}$$

where $\bar{K}_k$ is the number of different rewriting rules which match two pairs of k-grams (the same rule cannot trigger twice in k-gram substrings):

$$\bar{K}_k((\alpha_{s_1},\alpha_{t_1}),(\alpha_{s_2},\alpha_{t_2})) = \sum_{r\in R} \mathbb{1}_r(\alpha_{s_1},\alpha_{t_1})\mathbb{1}_r(\alpha_{s_2},\alpha_{t_2}) \tag{5.4}$$

with $\mathbb{1}_r$ the indicator function of rule r: 1 if r matches the pair of k-grams, 0 otherwise.
Computing K_k as defined in Equation 5.3 is obviously intractable. There is $\mathcal{O}((n-k+1)^4)$ terms in the sum, where n is the length of the longest string, and each term involves enumerating every rewriting rule in R.

5.1.3.2 Computing $\bar{K}_k$ in type-enriched kb-SRK

Enumerating all rewriting rules in Equation 5.4 is itself intractable: there are more than $|\Sigma|^{2k}$ rules without wildcards, where $|\Sigma|$ is conceivably the size of a typical lexicon. In fact, we just have to constructively generate the rules which substitute

their string patterns correctly to simultaneously produce both pairs of k-grams $(\alpha_{s_1}, \alpha_{t_1})$ and $(\alpha_{s_2}, \alpha_{t_2})$.

Let the operator $\otimes$ be such that $\alpha_1 \otimes \alpha_2 = ((\alpha_1[1], \alpha_2[1]), ..., (\alpha_1[k], \alpha_2[k]))$. This operation is generally known as *zipping* in functional programming. We use the function *CountPerfectMatchings* computed by Algorithm 1 to recursively count the number of rewriting rules matching both $(\alpha_{s_1}, \alpha_{t_1})$ and $(\alpha_{s_2}, \alpha_{t_2})$. The workings of the algorithm will make clearer why we can compute $\bar{K}_k$ with the following formula:

$$\bar{K}_k((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2})) = \text{CountPerfectMatchings}(\alpha_{s_1} \otimes \alpha_{s_2}, \alpha_{t_1} \otimes \alpha_{t_2}) \quad (5.5)$$

Algorithm 1 takes as input remaining character pairs in $\alpha_{s_1} \otimes \alpha_{s_2}$ and $\alpha_{t_1} \otimes \alpha_{t_2}$, and outputs the number of ways they can substitute aligned wildcards in a matching rule.

First (lines 2 and 3) we have the base case where both remaining sets are empty. There is exactly 1 way the empty set's wildcards can be aligned with each other: nothing is aligned. In lines 4 to 9, there is no source pairs anymore, so the algorithm continues to deplete target pairs as long as they have a common pattern type, i.e. as long as they do not have to substitute a wildcard. If a candidate wildcard is found, as the opposing set is empty, we cannot align it and we return 0. In the general case (lines 11 to 19), consider the first character pair (a_1, a_2) in the reminder of $\alpha_{s_1} \otimes \alpha_{s_2}$ in line 12. What follows in the computation depends on its types. Every character pair in $\alpha_{t_1} \otimes \alpha_{t_2}$ that can be paired through variable types with (a_1, a_2) (lines 15 to 19) is a new potential wildcard alignment, so we try all the possible alignment and recursively continue the computation after removing both aligned pairs. And if (a_1, a_2) does not need to substitute a wildcard because it has common pattern types (lines 13 and 14), we can choose to not create any wildcard pairing with it and ignore it in the recursive call.

This algorithm enumerates all configurations such that each character pair has a common pattern type or is matched 1-for-1 with a character pair with common variable types, which is exactly the definition of a rewriting rule in TESRK.

This problem is actually equivalent to counting the perfect matchings of the bipartite graph of potential wildcards. It has been shown intractable (Valiant, 1979) and Algorithm 1 is a naive recursive algorithm to solve it. In our implementation we represent the graph with its biadjacency matrix, and if our typing relations are independent of k, the function has a $\mathcal{O}(k)$ time complexity without including its recursive calls. The number of recursive calls can be greater than $k!^2$ which is the number of perfect matchings in a complete bipartite graph of $2k$ vertices. In our

Algorithm 1: Counting perfect matchings

```
1  CountPerfectMatchings(remS, remT)
   Data: remS: remaining char. pairs in source
   remT: remaining char. pairs in target
   graph: α_{s_1} ⊗ α_{s_2} and α_{t_1} ⊗ α_{t_2} as a bipartite graph, not added in the
   arguments to avoid cluttering the recursive calls
   ruleSet: Γ_p and Γ_v
   Result: Number of rewriting rules matching (α_{s_1}, α_{t_1}) and (α_{s_2}, α_{t_2})
2  if remS == ∅ and remT == ∅ then
3      return 1;
4  else if remS == ∅ then
5      (b_1, b_2) = remT.first();
6      if ∃γ ∈ Γ_p | b_1 ≈^γ b_2 then
7          return CountPerfectMatchings(∅, remT - {(b_1, b_2)});
8      else
9          return 0;
10 else
11     result = 0;
12     (a_1, a_2) = remS.first();
13     if ∃γ ∈ Γ_p | a_1 ≈^γ a_2 then
14         res += CountPerfectMatchings(remS - {(a_1, a_2)}, remT);
15     for (b_1, b_2) ∈ remT
16       | ∃γ ∈ Γ_v | a_1 ⇝^γ b_1 and a_2 ⇝^γ b_2 do
17         res += CountPerfectMatchings(
18             remS - {(a_1, a_2)},
19             remT - {(b_1, b_2)}
20         );
```

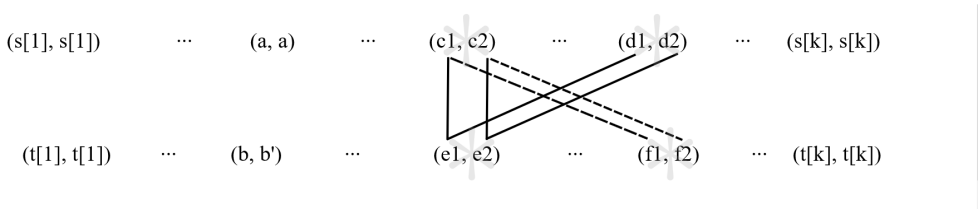

Figure 5.2: Bipartite graph of character pairs, with edges between potential wildcards

experiments on linguistic data however, we observed a linear number of recursive calls for low values of k, and up to a quadratic number for $k > 10$ –which is way past the point where the kernel becomes ineffective.

As an example, Figure 5.2 shows the zipped k-grams for source and target as a bipartite graph with $2k$ vertices and potential wildcard edges. Assuming that vertices (a, a) and (b, b') have common pattern types, they can be ignored as in lines 7 and 14. (c_1, c_2) to (f_1, f_2) however must substitute wildcards in a matching rewriting rule. If we align (c_1, c_2) with (e_1, e_2) in line 16, the recursive call will return 0 because the other two pairs cannot be aligned. A valid rule is generated if c's are paired with f's and d's with e's. This kind of choices is the main source of computational cost. This problem did not arise in the original kb-SRK because of the transitivity of its only type (*identity*). In type-enriched kb-SRK, wildcard pairing is less constrained.

5.1.3.3 Computing K_k

Even with an efficient method for computing $\bar{K}_k$, implementing K_k directly by applying Equation 5.3 remains impractical. The main idea is to efficiently compute a reasonably sized set $\mathbb{C}$ of elements $((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2}))$ which has the essential property of including all elements such that $\bar{K}_k((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2})) \neq 0$.
By definition of $\mathbb{C}$, we can compute efficiently

$$K_k((s_1, t_1), (s_2, t_2)) = \sum_{((\alpha_{s_1}, \alpha_{s_2}), (\alpha_{t_1}, \alpha_{t_2})) \in \mathbb{C}} \bar{K}_k((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2})) \tag{5.6}$$

There are a number of ways to do it, with a trade-off between computation time and number of elements in the reduced domain $\mathbb{C}$. The main idea of our own algorithm is that $\bar{K}_k((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2})) = 0$ if the character pairs $(a_1, a_2) \in \alpha_{s_1} \otimes \alpha_{s_2}$ with no common pattern type are not *all* matched with pairs $(b_1, b_2) \in \alpha_{t_1} \otimes \alpha_{t_2}$

such that $a_1 \overset{\gamma}{\rightsquigarrow} b_1$ and $a_2 \overset{\gamma}{\rightsquigarrow} b_2$ for some $\gamma \in \Gamma_v$. This is conversely true for character pairs in $\alpha_{t_1} \otimes \alpha_{t_2}$ with no common pattern type. More simply, character pairs with no common pattern type are mismatched and have to substitute a wildcard in a rewriting rule matching both $(\alpha_{s_1}, \alpha_{t_1})$ and $(\alpha_{s_2}, \alpha_{t_2})$. But introducing a wildcard on one side of the rule means that there is a matching wildcard on the other side, so we can eliminate k-gram quadruples that do not fill this wildcard inclusion. This filtering can be done efficiently and yields a manageable number of quadruples on which to compute $\bar{K}_k$.

Algorithm 2 computes a set $\mathbb{C}$ to be used in Equation 5.6 for computing the final value of kernel K_k. In our experiments, it efficiently produces a reasonable number of inputs. All maps in the algorithm are maps to multisets, and multisets are used extensively throughout. ***Multisets*** are an extension of sets where elements can appear multiple times, the number of times being called the ***multiplicity***. Typically implemented as hash tables from set elements to integers, they allow for constant-time retrieval of the number of a given element. Union ($\cup$) and intersection ($\cap$) have special definitions on multisets. If $\mathbb{1}_A(x)$ is the multiplicity of x in A, we have $\mathbb{1}_{A \cup B}(x) = max(\mathbb{1}_A(x), \mathbb{1}_B(x))$ and $\mathbb{1}_{A \cap B}(x) = min(\mathbb{1}_A(x), \mathbb{1}_B(x))$.

Algorithm 2: Computing a set including all elements on which $\bar{K}_k \neq 0$

Data: s_1, t_1, s_2, t_2 strings, and k an integer
Result: Set $\mathbb{C}$ which include all inputs such that $\bar{K}_k \neq 0$

Initialize maps $e^i_{s\to t}$ and maps $e^i_{t\to s}$, for $i \in \{1, 2\}$;
for $i \in \{1, 2\}$ **do**
    for $a \in s_i, b \in t_i \mid a \overset{\gamma}{\rightsquigarrow} b, \gamma \in \Gamma_v$ **do**
        $e^i_{s\to t}[a]$ += (b, γ); $e^i_{t\to s}[b]$ += (a, γ);
$w_{s\to t}, aP_t =$ OneWayInclusion($s_1, s_2, t_1, t_2, e^1_{s\to t}, e^2_{s\to t}$);
$w_{t\to s}, aP_s =$ OneWayInclusion($t_1, t_2, s_1, s_2, e^1_{t\to s}, e^2_{t\to s}$);
Initialize multiset res;
for $(\alpha_{s_1}, \alpha_{s_2}) \in aP_s$ **do**
    for $(\alpha_{t_1}, \alpha_{t_2}) \in aP_t$ **do**
        res += $((\alpha_{s_1}, \alpha_{s_2}), (\alpha_{t_1}, \alpha_{t_2}))$;
res = res $\cup w_{s\to t} \cup w_{t\to s}.map(swap)$;
return res;

`OneWayInclusion`$(s_1, s_2, t_1, t_2, e^1, e^2)$ Initialize map d multisets resWildcards, resAllPatterns;
for $(\alpha_{s_1}, \alpha_{s_2}) \in$ *kgrams*$(s_1) \times$ *kgrams*(s_2) **do**
    for $(b_1, b_2) \mid \exists \gamma \in \Gamma_v, (a_1, a_2) \in \alpha_{s_1} \otimes \alpha_{s_2}, (b_i, \gamma) \in e^i[a_i]\ \forall i \in \{1, 2\}$ **do**
        $d[(b_1, b_2)]$ += $(\alpha_{s_1}, \alpha_{s_2})$;
for $(\alpha_{t_1}, \alpha_{t_2}) \in$ *kgrams*$(t_1) \times$ *kgrams*(t_2) **do**
    for $(b_1, b_2) \in \alpha_{t_1} \otimes \alpha_{t_2} \mid b_1 \overset{\gamma}{\neq} b_2 \forall \gamma \in \Gamma_p$ **do**
        if *compatWkgrms not initialized* **then**
            Initialize multiset compatWkgrms = $d[(b_1, b_2)]$;
        compatWkgrms = compatWkgrms $\cap\ d[(b_1, b_2)]$;
    if *compatWkgrms not initialized* **then**
        resAllPatterns += $(\alpha_{t_1}, \alpha_{t_2})$;
    for $(\alpha_{s_1}, \alpha_{s_2}) \in$ *compatWkgrms* **do**
        resWildcards+=$((\alpha_{s_1}, \alpha_{s_2}), (\alpha_{t_1}, \alpha_{t_2}))$;
return (resWildcards, resAllPatterns);

Let us now comment on how the algorithm unfolds. In lines 1 to 4, we index characters in source strings by characters in target strings which have common variable types, and vice versa. It allows in lines 15 to 19 to quickly map a character pair to the set of opposing k-gram pairs with a matching –in the sense of variable types– character pair, i.e. potential aligned wildcards. In lines 20 to 28 we keep only the k-gram quadruples whose wildcard candidates (character pairs with no common pattern) from one side *all* find matches on the other side. We do not check for the other inclusion, hence the name of the function *OneWayInclusion*. At line 26, we did not find any character pair with no common pattern, so we save the k-gram pair as "all-pattern". All-pattern k-grams will be paired in lines 8 to 10 in the result. Finally, in line 11, we add the union of one-way compatible k-gram quadruples; calling swap on all the pairs of one set is necessary to consistently have sources on the left side and targets on the right side in the result.

5.1.4 Experiments

5.1.4.1 Systems

We experimented on three tasks: paraphrase identification, recognizing textual entailment and answer sentence selection. The setup we used for all experiments was the same save for the few parameters we explored such as: k, and typing scheme. We implemented 2 kernels, kb-SRK, henceforth simply denoted *SRK*, and the type-enriched kb-SRK, denoted *TESRK*. All sentences were tokenized and POS-tagged using OpenNLP (Morton et al., 2005). Then they were stemmed using the Porter stemmer (Porter, 2001) in the case of SRK. Various other pre-processing steps were applied in the case of TESRK: they are considered as types in the model and are detailed in Table 5.1. We used LIBSVM (Chang and Lin, 2011) to train a binary SVM classifier on the training data with our two kernels. The default SVM algorithm in LIBSVM uses a parameter C, roughly akin to a regularization parameter. We 10-fold cross-validated this parameter on the training data, optimizing with a grid search for f-score, or MRR for question-answering. All kernels were normalized using $\tilde{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)}\sqrt{K(y,y)}}$. We denote by "+" a sum of kernels, with normalizations applied both before and after summing. Following Bu et al. (Bu et al., 2012) experimental setup, we introduced an auxiliary vector kernel denoted *PR* of features named *unigram precision* and *recall*, defined in (Wan et al., 2006). In our experiments a linear kernel seemed to yield the best results. Our Scala implementation of kb-SRKs has an average throughput of about 1500 original kb-SRK computations per second, versus 500 type-enriched

Type	Typing relation on words (a, b)	Tool/resources
id	same surface form and tag	OpenNLP tagger
idMinusTag	same surface form	OpenNLP tokenizer
lemma	same lemma	WordNetStemmer
stem	same stem	Porter stemmer
synonym, antonym	words are [type]	WordNet
hypernym, hyponym	b is a [type] of a	WordNet
entailment, holonym		
ne	tagged with the same NE	BBN Identifinder
lvhsn	words at edit distance of 1	Levenshtein distance

Table 5.1: Types

kb-SRK computations per second on a 8-core machine. It typically takes a few hours on a 32-core machine to train, cross-validate and test on a full dataset.
Finally, Table 5.1 presents an overview of our types with how they are defined and implemented. Every type can be used both as a pattern type or as a variable type, but the two roles are different. Pattern types are useful to unify different surface forms of rewriting rules that are semantically equivalent, i.e. having semantically similar patterns. Variable types are useful for when the semantic relation between 2 entities across the same rewriting is more important than the entities themselves. That is why some types in Table 5.1 are inherently more fitted to be used for one role rather than the other. For example, it is unlikely that replacing a word in a pattern of a rewriting rule by one of its holonyms will yield a semantically similar rewriting rule, so *holonym* would not be a good pattern type for most applications. On the contrary, it can be very useful in a rewriting rule to type a wildcard link with the relation holonym, as this provides constrained semantic roles to the linked wildcards in the rule, thus *holonym* would be a good variable type.

5.1.4.2 Paraphrase identification

Paraphrase identification asks whether two sentences have the same meaning. The dataset we used to evaluate our systems is the MSR Paraphrase Corpus (Dolan and Brockett, 2005), containing 4,076 training pairs of sentences and 1,725 testing pairs. For example, the sentences *"An injured woman co-worker also was hospitalized and was listed in good condition."* and *"A woman was listed in good condition at Memorial's HealthPark campus, he said."* are paraphrases in this corpus. On the other hand, *"'There are a number of locations in our community,*

Paraphrase system	Accuracy	F-score
All paraphrase	66.5	79.9
Wan et al. (2006)	75.6	83.0
Bu et al. (2012)	76.3	N/A
Socher et al. (2011)	76.8	83.6
Madnani et al. (2012)	**77.4**	**84.1**
PR	73.5	82.1
SRK + PR	76.2	83.6
TESRK	76.6	83.7
TESRK + PR	**77.2**	**84.0**

Table 5.2: Evaluation results on MSR Paraphrase

which are essentially vulnerable,' Mr Ruddock said." and *"'There are a range of risks which are being seriously examined by competent authorities,' Mr Ruddock said."* are not paraphrases.

We report in Table 5.2 our best results, the system *TESRK + PR*, defined by the sum of PR and typed-enriched kb-SRKs with k from 1 to 4, with types $\Gamma_p = \Gamma_v = \{stem, synonym\}$. We observe that our results are state-of-the-art and in particular, they improve on the orignal kb-SRK by a good margin. We tried other combinations of types but it did not yield good results, this is probably due to the nature of the MSR corpus, which did not contain much more advanced variations from WordNet. The only statistically significant improvement we obtained was between *TESRK + PR* and our PR baseline ($p < 0.05$). The performances obtained by all the cited systems and ours are not significantly different in any statistical sense. We made a special effort to try to reproduce as best as we could the original kb-SRK performances (Bu et al., 2012), although our implementation and theirs should theoretically be equivalent.

Figure 5.3 plots the average number of recursive calls to CountPerfectMatchings (algorithm 1) during a kernel computation, as a function of k. Composing with log_k, we can observe whether the empiric number of recursive calls is closer to $\mathcal{O}(k)$ or $\mathcal{O}(k^2)$. We conclude that this element of complexity is linear for low values of k, but tends to explode past $k = 7$. Thankfully, counting common rewriting rules on pairs of 7-to-10-grams rarely yields non-zero results, so in practice using high values of k is not interesting.

Figure 5.4 plots the average size of set $\mathbb{C}$ computed by algorithm 2, as a func-

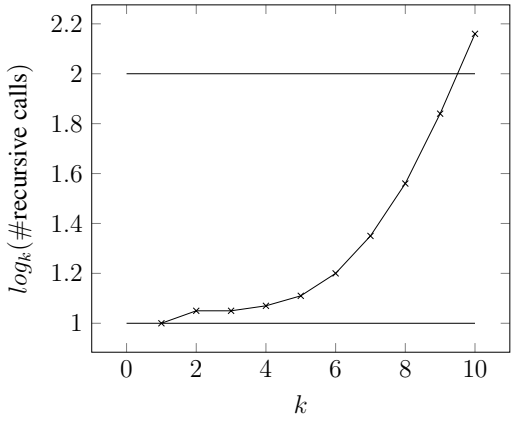

Figure 5.3: Evolution of the number of recursive calls to CountPerfectMatchings with k

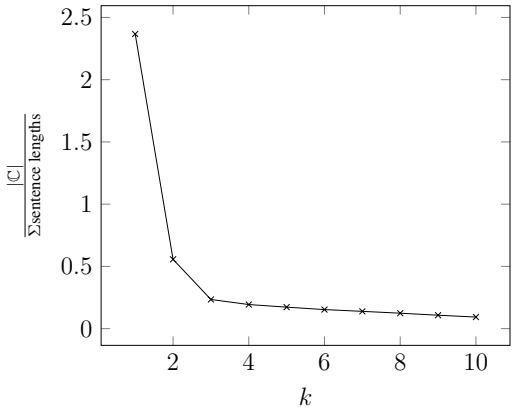

Figure 5.4: Evolution of the size of $\mathbb{C}$ with k

RTE system	Accuracy
All entailments	51.2
Heilman and Smith (2010)	62.8
Bu et al. (2012)	65.1
Zanzotto et al. (2007)	65.8
Hickl et al. (2006)	**80.0**
PR	61.8
TESRK (All)	62.1
SRK + PR	63.8
TESRK (Syn) + PR	64.1
TESRK (All) + PR	66.1

Table 5.3: Evaluation results on RTE-3

tion of k (divided by the sum of lengths of the 4 sentences involved in the kernel computation). We can observe that this quantity is small, except for a peak at low values of k, which is not an issue because the computation of $\bar{K}_k$ is very fast for those values of k.

5.1.4.3 Recognizing textual entailment

Recognizing Textual Entailment asks whether the meaning of a sentence *hypothesis* can be inferred by reading a sentence *text*. The dataset we used to evaluate our systems is RTE-3. Following similar work (Heilman and Smith, 2010; Bu et al., 2012), we took as training data (text, hypothesis) pairs from RTE-1 and RTE-2's whole datasets and from RTE-3's training data, which amounts to 3,767 sentence pairs. We tested on RTE-3 testing data containing 800 sentence pairs. For example, a valid textual entailment in this dataset is the pair of sentences *"In a move widely viewed as surprising, the Bank of England raised UK interest rates from 5% to 5.25%, the highest in five years."* and *"UK interest rates went up from 5% to 5.25%."*: the first entails the second. On the other hand, the pair *"Former French president General Charles de Gaulle died in November. More than 6,000 people attended a requiem mass for him at Notre Dame cathedral in Paris."* and *"Charles de Gaulle died in 1970."* does not constitute a textual entailment.

We report in Table 5.3 our best results, the system *TESRK (All) + PR*, defined by the sum of PR, 1b-SRK and typed-enriched kb-SRKs with k from 2 to 4, with types $\Gamma_p = \{$stem, synonym$\}$ and $\Gamma_v = \{$stem, synonym, hypernym, hyponym,

entailment, holonym$\}$. Our results are to be compared with systems using techniques and resources of similar nature, but as reference the top performance at RTE-3 is still reported. This time we did not manage to fully reproduce Bu et al. 2012's performance, but we observe that type-enriched kb-SRK greatly improves upon our original implementation of kb-SRK and outperforms their system anyway. Combining TESRK and the PR baseline yields significantly better results than either one alone ($p < 0.05$), and performs significantly better than the system of (Heilman and Smith, 2010), the only one which was evaluated on the same three tasks as us ($p < 0.10$). We tried with less types in our system *TESRK (Syn) + PR* by removing all WordNet types but synonyms but got lower performance. This seems to indicate that rich types indeed help capturing more complex sentence rewritings. Note that we needed for $k = 1$ to replace the type-enriched kb-SRK by the original kernel in the sum, otherwise the performance dropped significantly. Our conclusion is that including richer types is only beneficial if they are captured within a context of a couple of words and that including all those variations on unigrams only add noise.

5.1.4.4 Answer sentence selection

Answer sentence selection is the problem of selecting among single candidate sentences the ones containing the correct answer to an open-domain factoid question. The dataset we used to evaluate our system on this task was created by (Wang et al., 2007) based on the QA track of past Text REtrieval Conferences (TREC-QA)[1]. The training set contains 4718 question/answer pairs, for 94 questions, originating from TREC 8 to 12. The testing set contains 1517 pairs for 89 questions. As an example, a correct answer to the question:

(38) What do practitioners of Wicca worship?

is

(39) An estimated 50,000 Americans practice Wicca, a form of polytheistic nature worship.

On the other hand, the answer candidate:

(40) When people think of Wicca, they think of either Satanism or silly mumbo jumbo.

[1] Available at http://nlp.stanford.edu/mengqiu/data/qg-emnlp07-data.tgz

System	MAP	MRR
Random baseline	0.397	0.493
Wang et al. (2007)	0.603	0.685
Heilman and Smith (2010)	0.609	0.692
Wang and Manning (2010)	0.595	0.695
Yao et al. (2013)	0.631	0.748
Yih et al. (2013) LCLR	**0.709**	**0.770**
IDF word-count (IDF)	0.596	0.650
SRK	0.609	0.669
SRK + IDF	0.620	0.677
TESRK (WN)	0.642	0.725
TESRK (WN+NE)	0.656	0.744
TESRK (WN) + IDF	0.678	0.759
TESRK (WN+NE) + IDF	0.672	**0.768**

Table 5.4: Evaluation results on QA

is incorrect. Sentences with more than 40 words and questions with only positive or only negative answers were filtered out (Yao et al., 2013a). The average fraction of correct answers per question is 7.4% for training and 18.7% for testing. Performances are evaluated as for a re-ranking problem, in term of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). We report our results in Table 5.4. We evaluated several combinations of features. *IDF word-count* (IDF) is a baseline of IDF-weighted common word counting, integrated in a linear kernel. Then we implemented SRK and TESRK (with k from 1 to 5) with two typing schemes: *WN* stands for $\Gamma_p = \{$stem, synonym$\}$ and $\Gamma_v = \{$stem, synonym, hypernym, hyponym, entailment, holonym$\}$, and *WN+NE* adds type ne to both sets of types. We finally summed our kernels with the IDF baseline kernel. We observe that types which make use of WordNet variations seem to increase the most our performance. Our assumption was that named entities would be useful for question answering and that we could learn associations between question type and answer type through variations: NE does seem to help a little when combined with WN alone, but is less useful once TESRK is combined with our baseline of IDF-weighted common words. Overall, typing capabilities allow TESRK to obtain way better performances than SRK in both MAP and MRR, and our best system combining all our features is comparable to state-of-the-art

systems in MRR, and significantly outperforms *SRK + IDF*, the system without types ($p < 0.05$).

5.1.5 Discussion

Lodhi et al. (Lodhi et al., 2002) were among the first in NLP to use kernels: they apply *string kernels* which count common subsequences to text classification. Sentence pair classification however require the capture of 2 types of links: the link between sentences within a pair, and the link between pairs. Zanzotto et al. (Zanzotto et al., 2007) used a kernel method on syntactic tree pairs. They expanded on graph kernels in (Zanzotto et al., 2010). Their method first aligns tree nodes of a pair of sentences to form a single tree with placeholders. They then use *tree kernel* (Moschitti, 2006) to compute the number of common subtrees of those trees. Bu et al. (Bu et al., 2012) introduced a string rewriting kernel which can capture at once lexical equivalents and common syntactic dependencies on pair of sentences. All these kernel methods require an exact match or assume prior partial matches between words, thus limiting the kind of learned rewriting rules. Our contribution addresses this issue with a type-enriched string rewriting kernel which can account for lexico-semantic variations of words. Limitations of our rewriting rules include the impossibility to skip a pattern word and to replace wildcards by multiple words.

Some recent contributions (Chang et al., 2010; Wang and Manning, 2010) also provide a uniform way to learn both intermediary representations and a decision function using potentially rich feature sets. They use heuristics in the joint learning process to reduce the computational cost, while our kernel approach with a simple sequential representation of sentences has the benefit of efficiently computing an exact number of common rewriting rules between rewriting pairs. This in turn allows to precisely fine-tune the shape of desired rewriting rules through the design of the typing scheme.

A strong point of our work is its evaluation on 3 tasks, with state-of-the-art results in two of them. To our knowledge, the only other system evaluated on the same three tasks is that of Heilman and Smith (2010), which we outperform significantly on all tasks. Let us look closer to the task of RTE, where we are far behind the top result: Hickl et al. (2006) has an accuracy of 80.0% on RTE 3, while our system only tops at 66.1%. Already mentioned in our literature review, Hickl et al. (2006) include many features we do not have, like extra preprocessing: semantic parsing, coreference resolution, modality, polarity, factivity, or extra resources, like their paraphrase database. The inclusion of paraphrases as

a resource is something this current kernel method is not capable of, but something we thought about and would like to explore going forward. The problem is that our current types are restricted to single words, and a paraphrase would be a kind of typing spanning several words. This calls for either a redesign of the type system and as such, of the entire computation method, or maybe more interestingly, for a clever combination of several types to encode the application of a paraphrase on several words.

We can also identify some clear limitations of our system. It is expensive in term of computing power – it could not run standard evaluations in acceptable times on a current personal computer – and does not scale well with the number of training examples. This is a huge downside at a time where data is becoming more and more available. Paradoxically enough, the way the kernel is defined also implies a good amount of training examples, otherwise rewriting rules occur too sparsely to be really learned by the model.

In the next section, we look to keep this idea of rewritings backed up by lexical-semantic resources, but we instead use a tree edit model instead of a kernel. From the edit sequences, we extract features to be used in a classic logistic regression classifier, which is much more scalable. The edit-based features, heuristics on trees and the resources we can include allow this method to be effective even with few training examples.

5.2 Tree Edit Beam Search

Tree-edit methods are easy to conceptualize and run: a set of elementary operations on tree is applied iteratively to a *source* tree to transform it into a *target* tree. Generally, to solve a decision problem, *edit sequences* – i.e. a succession of edit operations – are extracted and the related features are used in a machine learning classifier. But the crux of such methods is not in the definition of the edit operations, nor the definition of the edit sequence features, as these are rather straight-forward. The hard part is to know which edit to apply at which moment, among a lot of possible transformations.

In this section, we present an algorithm called *Tree Edit Beam Search*. "Beam search" precisely refers to the method for picking the right edit operations. The algorithm starts from a source tree and applies a complete set of classic edits to form as many new trees. As this is usually too high a number of trees to handle,

Edit operation	Description
Delete(d: Tree)	Delete the node *d* and replace it with its children.
Insert(i: Word, p: Tree)	Insert the word *i* under its new parent *p*.
Rename(t: Tree, w: Word)	Replace the word attached to the node *t* with *w*.
Move(m: Tree, op: Tree, np: Tree)	Move the subtree *m* from under *op* to under *np*.

Table 5.5: Edit operations on trees

for the next step of editing we only keep the most promising trees. A tree kernel evaluates the distance from one of these trees to the target tree. We extract a limited number of edit sequences from the source tree to the target tree. Finally, features are computed and used in a logistic regression classifier.

This method has been evaluated on the task of reading comprehension tests, "Entrance exams" at CLEF 2015 and published at the related conference (Gleize and Grau, 2015a). We detail in this section the features used, and especially the resources we leveraged. Our results at CLEF 2015 are mentioned, but the full system is further detailed in Chapter 6.

5.2.1 Tree edit model

5.2.1.1 The model

The basis of this method is the tree edit model. Edit operations are applied one at a time, to produce a new tree per edit. The process is then iterated on the new structures, and the list of consecutive edit operations applied in this way constitutes the edit sequence. Figure 5.5 shows an example of an edit sequence. Edit operations are generally simple transformations of the tree applied locally to one or at most 2 nodes. The set we use is very classic, and is shown in Table 5.5. The "delete", "insert" and "rename" operations are the minimal operations in any edit model. The "move" operation is more unusual and allows to make seemingly important changes to the sentence just by attaching one node from its former parent to a new parent. Because of how trees work, a move operation can indeed change the position of a large span of words (all the descendants of the moved node). We note that the algorithm does not use the dependency labels that are found on the relations between nodes in the tree. However, the features will use them.

It is important to keep in mind that we want to apply a lot of those edit operations to produce a lot of different trees. We will be able to discard most of them

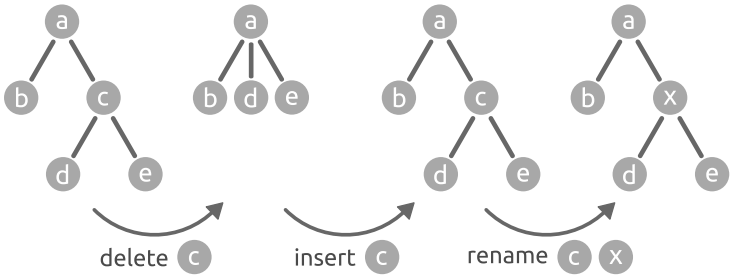

Figure 5.5: Examples of successive edit operations

eventually, as shown in the next part, but a good model should be able to handle as most edits as possible, to not miss any useful edit that might lead easily to the target tree. This implies the necessity of an efficient memory structure.

5.2.1.2 The implementation

We initially thought about encoding all edits in a single tree with "choice" nodes, nodes that would imply a choice between several edits or the original node at this position. However, we realized that it was not possible to do so in a simple way: indeed, subsequent edits depend on the edits applied beforehand, and as such, the order in which they are applied cannot be determined freely. The obvious examples are that we cannot delete a node twice, nor rename a node we deleted before, nor delete a node that we did not insert before.

So we had to represent each tree in a very efficient way. What we do is represent the children of a node as an immutable indexed sequence called a *vector*. Vectors are implemented as tries – another tree structure – with a high branching factor (the branching factor of a tree or a graph is the number of children at each node). These tries must not be confused with our trees of words however, they only are the background implementation of the children. Every trie node contains up to 32 elements of the vector or contains up to 32 other trie nodes. Vectors with up to 32 elements can be represented in a single node. Vectors with up to 32 * 32 = 1024 elements can be represented with a single indirection. For all vectors of reasonable size, an element selection involves up to 5 primitive array selections. So element access is effectively constant time. Updating an element in the middle of a vector can be done by copying the node that contains the element, and every node that points to it, starting from the root of the trie. This means that a functional update creates between one and five nodes that each contain up to 32 elements or subtrees. While this is certainly more expensive than an in-place update

representing the edit operation in a tree as we intended before, this is much more efficient than a full copy of the tree. Considering we only work on trees with small branching factors (no linguistic tree node has more than 32 children), applying an edit operation is pretty much done in constant time and space, and especially does not depend on the size of the tree and the number of already applied edits. Concretely, we did not have to implement the vector structure, as Vectors are one of the standard data structures in the programming language we use, Scala; we just had to identify that it indeed fulfills our needs nicely.

We have defined the edit operations and have an efficient way of representing them in memory, but we still need to pick the ones to explore as it is too expensive in time to consider all possible edit sequences. As we are technically searching for the target tree, exploring many other trees along the way, this problem can be seen as a search on graph.

5.2.2 Beam search

Starting from a tree at any point in the algorithm, there are many possible edits to apply: each node can be deleted, each node can be renamed to any arbitrary label, any arbitrary node can be inserted and any subtree can be moved to any remaining node. We will apply simple heuristics to limit the number of edits to consider at each step (for example, we do not need to insert nodes that do not help reach the target tree, and we do not have to rename or delete a node if it is already present in the target tree), but a lot of choices remain. As the number of trees increases exponentially as we do more edit steps, we rapidly become unable to handle the number of possible trees in the search space. Thus, we need some kind of efficient exploration method.

Beam search is an optimization of best-first search that reduces its memory requirements. It uses breadth-first search to build its search space. At each step of the search, we generate all possible edits of the trees at the current step, sorting them in increasing order of some *heuristic cost function*. However, we only store a predetermined number of best trees at each level, called the beam width. Only those trees are edited next and the rest is discarded. This method allows to fine-tune via the beam width the probability to find useful edit sequences and the memory and time costs. Figure 5.6 shows an example of how beam search reduces the search space. On the figure, the circles represent each of the trees, with the root circle being the source tree. Using a beam width of 2, at each step we only keep the 2 most promising trees. We see that each edit tree is ranked using the heuristic cost function (that we will present below), and only the top 2 trees

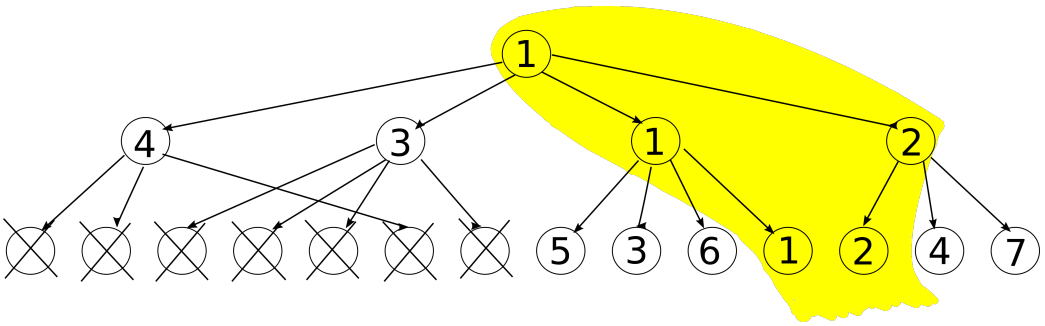

Figure 5.6: Beam search with a beam width of 2

are further edited. Crossed out circles are all the trees that the algorithm did not have to explore at any point. The effective search space is reduced to the narrow colored band, the so-called "beam".

In practice, parent trees actually re-enter the pool of possible trees to keep in the beam-search. If their heuristic cost is somehow lower than that of some of their children, they will have priority in the next beam search step. This is much less easy to represent graphically, and one can imagine we can add another edit operation that alters nothing so as to keep the original trees in the search.

The beam width is obviously a parameter of this search procedure and has to be chosen carefully, but what is much more important to address is the heuristic cost function. In our case, it has to measure the distance from a tree to the target tree, so as to evaluate how far we still have to go. The closest trees to the target tree will have priority in the beam search. We use the *Partial Tree Kernel* as the heuristic. It is defined by Moschitti (2006) to compute a similarity between trees. As a tree kernel, it classically computes the number of common subtrees between 2 trees, but this particular version of the tree kernel is adapted to n-ary trees, which is what dependency structures are. The kernel computation is normalized with $\tilde{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)}\sqrt{K(y,y)}}$, for K the kernel and x, y the trees.

We limit the number of edits at each step by only testing the ones likely to help reach the target tree.

- Only insert/rename a word as many times as needed to match its count in the target tree
- Only move a node under a new parent so that the same parent-child relation is present in the target tree

- Delete a word only as long as there are enough remaining occurrences in the tree to match the number of occurrences in the target tree

The beam search runs until 10 different edit sequences leading to the target tree are found. We note that there is always such an edit sequence: delete all the nodes of the source tree, then insert all the nodes of the target tree.

5.2.3 Feature extraction

The goal is to classify an edit sequence with a machine-learned classifier. We experiment on reading comprehension tests, where the target tree is the tree of answer choice. The only gold standard annotation available is if the answer is correct or not. The design of features is thus primordial. First we describe our pre-processing tools, then the resources we used, and finally we present the complete set of features.

5.2.3.1 Pre-processing

We use Stanford CoreNLP as the main Natural Language annotation tool. Each sentence from the document, questions or answer choices is tagged with Part-Of-Speech (Toutanova et al., 2003) and syntactically parsed (Klein and Manning, 2003). In addition, a coreference resolution system (Recasens et al., 2013) is applied on the whole document as well as question-answer pairs. We add to this coreference resolution process manual rules specific to the task ("Entrance exams") and its dataset: we replace first person pronouns in non-dialogue context with "the author" or "the writer", depending on which is used in the questions.

Also, contrary to previous methods, we have to handle multi-sentence texts. The answer choice is a single sentence, but the relevant passages in the document can span several sentences. We fuse the trees of the passages together by linking their roots with a *followed-by* arc which materializes in the single remaining graph that a sentence is followed by another in the passage.

5.2.3.2 Resources

First, we use the following WordNet relations for lexical-semantic variations: synonymy, antonymy, hypernymy and hyponymy[2]. These are not added right away

[2]We already described WordNet in this dissertation so we will not dwell on it further in this section

to the source or target trees before the tree-edit run, but will be considered when computing the features.

We also use *ConceptNet* (Liu and Singh, 2004) to enrich the source tree. ConceptNet is a semantic triplet base containing relations about common-knowledge of the world, designed to be used especially for machine understanding of text written by people. It is built from nodes representing words or short phrases of natural language, and labeled relationships between them (the nodes are called "concepts" for tradition, but they would be better known as "terms".) For example, ConceptNet contains everyday basic knowledge, like *MotivatedByGoal(learn, knowledge)*: you would learn because you want knowledge. It also contains cultural knowledge, like *UsedFor(saxophone, jazz)*: a saxophone is used for jazz. Our assumption is that understanding the documents in the Entrance exams corpus requires a lot of human common-sense, easily acquired by human readers of that level, but difficult to grasp for computers. So we want to enrich the text with relations that attempt to fill that gap.

Concepts from ConceptNet are mainly single words, like “saxophone” or “jazz”, so they are easy to link to our original tree. However, it is not easy to integrate *relations* to our graph, because they have labels that are potentially composed of several words, like UsedFor or MotivatedByGoal. We could split those labels into words and use those in the graph, but we preferred attaching to the original graph the parse tree of the *surfaceText* element of the relations. Surface texts are the original natural language text that expressed the statement, like “a saxophone is used for jazz”. We attach the parse tree of these sentences to any concept whose head word is in the original graph, prior to the tree-edit run. We only retrieve from ConceptNet relations that are indicative of an entailment relation of any kind, namely: IsA, PartOf, MemberOf, UsedFor, CapableOf, Causes, HasPrerequisite, MotivatedByGoal, Desires.

The source tree is now enriched by a lot of most likely useless relations. However, the tree-edit model will delete these naturally. The features presented in the following section will record when we use a ConceptNet addition in a more interesting way, like moving it in another part of the graph, or renaming one of its words.

5.2.3.3 Complete set of features

Most features are counts of specific edit unigrams or bigrams in the edit sequence, and are summarized in Table 5.6. Pre-processing informations that were not used in the beam search are used at this point, like dependency relations in the parse

tree, coreferences, and whether what we edit was part of the ConceptNet additions or can be linked in WordNet.

This feature set is interesting when compared to the "features" of the typed string re-writing kernel presented in Section 5.1 (p. 117). The feature vectors in the kernel method are intentionally of very high dimension: they represent the number of occurrencs of every possible k-gram rewriting rule. Our inability to represent in a concrete way such a feature set is why we use a kernel in the first place. In this case however, the features are of low dimension. As we have access to a small amount of training data, a tree-edit model felt like a natural way to both capture structured rewriting processes on sentences, and to learn how to use all these rules and resources. In the next section, we briefly describe our experiments and results to see if this guess turned out to be successful.

5.2.4 Experiments

We used the Tree Edit Beam Search as part of our submission in the Entrance Exams task of CLEF 2015 (Rodrigo et al., 2015).

5.2.4.1 Data

Our data consist of the trial and test sets at CLEF 2015 Question Answering Track, Task 2: Entrance Exams. The trial data is composed of the test sets at CLEF 2013 and 2014, each containing a series of 12 texts, and for each of them, 4 to 6 multiple-choice questions to answer, for about 120 questions in total. In the 2015 test set, there are 19 documents, and a total of 89 questions. There are 4 answer choices possible for each of the questions. This corpus has been extracted from the Tokyo University Entrance Exam in English as a foreign language.

5.2.4.2 A note on the complete system

The complete system is described in the last chapter of this thesis, but here is a summary of what we add on top of the Tree Edit Beam Search for it to work on this task. We implemented a passage ranking method, and the best passages are turned into trees as input for the tree edit method. We extract features from the resulting edit sequences as described previously. We train 2 classifiers, one to validate the answer choice and one to invalidate the answer choice: the features stay the same, the training data are the ones to vary. We test with 2 different kinds of classifiers: random forest (Liaw and Wiener, 2002) and logistic regression. The

Feature	Description
editTotal	Total number of edits in the sequence
deleteTotal deleteVerb deleteNoun deleteProperNoun deleteSubject deleteObject deleteRoot deleteNegation deleteConceptNet	Number of total delete edits, edits which delete a verb, a noun, a proper noun, a subject (indicated by the subj Stanford dependencies), an object, the root of the tree, a negation (indicated by the neg dependency), and something added to the graph through ConceptNet
insertTotal insertVerb insertNoun insertProperNoun insertNegation	Analogous to the above, for insert edits
renameTotal renameVerb ... renameSyn renameAnt renameHypHyp renameStrongWordVectorSim renameCoref renameNonCoref	Analogous, for rename edits + edits which rename a word into its synonym/antonym/hypernym/hyponym in Wordnet, edits which rename a word into another with strong word vector similarity, edits which rename a pronoun into its referent, ...
moveTotal moveVerb ... moveConceptNet moveMoreThan2Nodes	Analogous to the above, for move edits + edits which move more than 2 nodes
All bigram combinations of the above	Number of pairs of the successive given edits in the sequence
dependencyEditSequence	Number of pairs of successive edits applied to 2 nodes in a dependency relation
originalTotal originalVerb ...	Fraction of the original words, verbs, nouns, proper nouns, that was **not** edited in the sequence

Table 5.6: Features of an edit sequence

same kind of classifier is used for both validation and invalidation, it just means that the machine learning algorithm is seen as a kind of parameter of our system. We pick the answer choice with the best score, taking into account its validation and invalidation classification.

5.2.4.3 Results

Table 5.7 shows our results on the test data. We officially submitted to the evaluation task only two runs, one with random forest as the learning method and one with logistic regression. With an accuracy of 0.36, the random forest run obtained the second best result at the evaluation, behind the run of Laurent et al. (2015), at 0.58. We remind that the random baseline, picking among the 4 answer choices at random, get 0.25.

We performed further tests only afterward. These tests are ablation tests on the resources used by the system: WordNet and ConceptNet. Our original system only performed the enrichment of the tree by ConceptNet relations on passage trees, not on the trees representing the answer choices. Our intention was for the answer choice to stay in its original form, so as to not introduce unnecessary noise in the heuristic cost function of our method. We test the same enrichment on the tree of the answer choice, in addition to applying it on the passages ("ConceptNet on answer too"). Then we test by removing ConceptNet relations altogether. Finally, we also remove WordNet, which means that our system only relies on its pre-processing tools in this run.

First, we notice that adding ConceptNet relations to the answer directly was not a good idea, as the performance drops dramatically. The rest of the results are less clear-cut. Removing WordNet does seem to affect the system negatively, but not as much as one might think. Removing ConceptNet also decreases the performance by a small amount.

5.2.5 Discussion

Tree-edit models offer a lot of freedom in the choice of rewriting rules that can be applied to a sentence. In comparison, the first work we presented in this chapter seems to restrict its rules more, although types somewhat alleviate the problem. But this freedom of choice also leads to a huge search space to explore, forcing the system to use some kind of search heuristic (like our beam search). In contrast, our first method was computing an exact measure of the number of common rewriting

	Random forest	Logistic regression
Questions answered	89	89
Errors	57	61
Accuracy	0.360	0.315
- ConceptNet on answer too		
Accuracy	0.292	0.270
- Without ConceptNet		
Accuracy	0.348	0.303
- Without WordNet		
Accuracy	0.337	0.292

Table 5.7: Results on Entrance exams at CLEF 2015

rules between sentence pairs. In both cases, it seems quite difficult to analyze in a qualitative manner which types of rules are really learned by the models.

Tree edit models are popular in the literature, as shown in our literature review, but the one contribution that is the most similar to us is that of Heilman and Smith (2010). They use roughly the same set of edit operations. Where our methods differ is in the choice of search algorithm and the features. They implement a greedy best-first search, while we use a beam search. The two algorithms are actually not that different, but the greedy best-first search filters out more potentially good edit sequences to focus first on the best apparently available (that is the "greedy" aspect). We include ConceptNet as an extra resource, where they only use WordNet. This richer feature set does seem to help a bit in our task.

5.3 Conclusion

This chapter presented sentence rewriting methods which integrate the use of background resources and machine learning.

The first, the type-enriched string rewriting kernel, deals with a string model of the sentence, but still can capture local structure through its definition of the n-gram rewriting rules. We extended an existing model to accept "types", which really are used to include lexical-semantic variations of words and extra layers of annotation. We show that the cost in space and time complexity of this addition is manageable, and show good results on three different CQI-related tasks. Its limitations are however as clear as its benefits: as a kernel method, it cannot scale directly to huge training datasets.

The second method is designed to fulfill some of the requirements the first method by-passes. Most of the heavy lifting is performed by a tree edit beam search algorithm, which defines a reduced search space of edited trees to go from a source tree (for example, a Text) to a target tree (a Hypothesis). The machine learning itself only uses a simple feature set extracted from the edit sequences, and we can choose the algorithm so that the method is very scalable. We evaluate this method on reading comprehension tests and obtained good results, even if the benefit of the inclusion of resources is not obvious.

This chapter did not deal with the full magnitude of the Contextually Queried Inference problem. Our systems only ever deal with two sentences, and at best a passage and a sentence. Our first system is only as of yet evaluated on simple textual inference tasks involving no context. In the next chapter, we focus solely on the task of reading comprehension tests, which was identified early on in this thesis as one of the most complete, yet easy-to-evaluate textual inference problem.

Chapter 6

Our approach to a complete application of the CQI task: reading comprehension tests

Contextually Queried Inference is our idea of a generic and complete textual inference task. In Chapter 3, we showed that a lot of tasks are certainly encompassed by CQI. But solving CQI is not necessary to solve RTE, paraphrase identification or answer sentence selection as they were defined previously. In this last chapter, we are interested in a task fitting most completely the definition of CQI: reading comprehension tests. These tests are usually taken by human students, at the entrance of universities or to test their knowledge in a foreign language. In the form of multiple-choice questions tied to a text, they are easy to evaluate and can test abilities ranging from lexicon size to complex inference.

In the first part, we describe a corpus built throughout the three years of the Entrance Exams task at CLEF 2013-2015. This corpus is quite unique in term of the type of linguistic and inferential phenomena it covers. We added to it manual annotations, which we intend to use as training data in our submission to the evaluation campaign.

In the second part, we present our approach to this task, based on validation and invalidation. Each question presents four choices, and we quickly realized that it is often easier to invalidate several incorrect answers, then decide on the correct one, rather than validate an answer choice in a vacuum, independently from the others. Our first method uses manual rules and does not meet much success, but the second method leverages a more sizable training corpus to learn two classifiers, one for validation and one for invalidation. This last method obtains

the second best result at CLEF 2015's Entrance Exams evaluation.

6.1 The Entrance Exams task at CLEF

In this section, we describe the Entrance Exams task at CLEF. As we have already mentioned it several times in this thesis, we focus more on the nature of the corpus itself, rather than the modalities of the evaluation. In particular, we present an extra annotation layer that we added to the texts: relevant passages for all the answer choices.

6.1.1 Task definition

Entrance exams is an evaluation task which ran at CLEF in 2013, 2014, 2015. The participants' performance steadily increased throughout these years, as participants gained insight on the data and built resources and modules. Nonetheless, it remains one of the hardest task in natural language understanding.

At the basis of this evaluation is a text, from 300 to 800 words long. It is provided with 3 to 6 multiple-choice questions to answer. Each question has 4 answer choices, and exactly one is the correct one. CLEF 2013 and 2014 datasets had each 12 texts and 60 questions. CLEF 2015 has 19 texts and 89 questions. An example of question from the 2015 corpus, with the relevant passage of the text, is provided below. These tests come from Japanese entrance exams at Tokyo University and evaluate English as a foreign language. The exams are created by the Japanese National Center for University Admissions Tests. The specific "Entrance Exams" corpus used in the evaluation is provided by NII's Todai Robot Project (Fujita et al., 2014) and NTCIR RITE (Matsuyoshi et al., 2014).

Text:
It was early morning. Peter Corbett helped Mark Wellman out of his wheelchair and onto the ground. They stood before El Capitan, a huge mass of rock almost three-quarters of a mile high in California's beautiful Yosemite Valley. It had been Mark's dream to climb El Capitan for as long as he could remember. But how could a person without the use of his legs hope to try to climb the highest vertical cliff on earth?
[...]
Question: *What had Mark Wellman long desired to do?*

1. **To accomplish one of the most difficult rock climbs in the world. (correct answer)**
2. To be the first to conquer El Capitan.
3. To climb the highest mountain in California.
4. To help his friend Peter climb El Capitan.

The goal is to correctly answer as many questions as possible, and the evaluation measures are simple. The accuracy is the main measure, i.e. the fraction of correctly answered questions. Systems are also evaluated according to their c@1, defined in equation 6.1.

$$C@1 = \frac{1}{n}(n_R + n_U \frac{n_R}{n}) \tag{6.1}$$

with n the total number of questions, n_R the number of correctly answered questions, n_U the number of unanswered questions. This measure favors systems who would be able to decide to purposely answer only some questions and not others, but so far, over the 3 years the evaluation has run, no system has been able to manifest that capability.

6.1.2 The corpus

In this section, we explain the characteristics of the reading test corpus, for each of its three types of component: text, question and answer choice.

6.1.2.1 Text

Texts are quite short, ranging from 300 words to 800 words. Generally, the smaller the text, the fewer the questions provided with it. We start this section with a few examples of the first few sentences of texts. This choice of examples is meant to show the variety in content and tone.

(41) Several years ago, certain scientists developed a way of investigating the nature of the atmosphere of the past by studying air caught in the ice around the North or South Pole. According to their theory, when snow falls, air is trapped between the snowflakes. The snow turns to ice with the air still inside.

(42) Butterflies are insects as familiar to us as dragonflies. Many of us remember chasing them in the countryside or seeing them pinned neatly in boxes in museums. There are many people who collect butterflies because they are fascinated by their beauty and variety. Butterfly shapes have also been used for patterns on kimono for a long time. Nowadays butterflies are usually considered to be objects of beauty.

(43) Old Fred Ford had gone to live with his daughter, Kate, and her family shortly after his wife, Mary, died. Kate was an energetic woman who expected people always to be doing something, and she found plenty of jobs for Fred to do. This made him feel part of the household, but now he really wanted to be able to sit and reflect on the events of his life.

(44) On his first day of elementary school, Johnny came home, banged the front door open, threw his cap on the floor, and shouted, "Isn't anybody here?" At dinner he spoke rudely to his father and spilled his baby sister's milk. "How was school today?" I asked casually. "All right" Johnny said. "Did you learn anything?" his father asked. "I didn't learn nothing," he said.

(45) The first time I met him, everybody seemed to think that he was crazy and very dangerous. However, I was fascinated with him and gradually a new friendship was born between us. It all began on the day I visited a yacht in Newport Harbor. A friend of mine, Richard, owned the yacht. He and his wife had two female relatives staying with them during the summer.

At first glance, these examples denote two very different types of texts. The first and second samples are the beginning sentences of informative texts, maybe

even newspaper articles. The texts of this type are mostly texts of pop-science or explaining social trends. For example, there are no texts about politics or history. Well-known figures, personalities or groups are only rarely mentioned.

The last three examples are the beginning sentences of narrative texts, most likely fiction novels. Some are third-person narratives, like the third example, and the others are first-person narratives. The text can also contain a heavy amount of dialogue, as shown in the fourth example.

These texts share in common one very important aspect: they do not require knowledge on named entities or important events or locations to be understood. Commonsense knowledge is enough to understand the topic of the text and most likely its development. We remind the reader that these texts are used in exams of English as a foreign language. Thus, it seems natural for the texts to remain neutral on any cultural knowledge and focus more on day-to-day knowledge, as this form is likely to be universally accessible to any student who takes the test. This allows the evaluation to put an emphasis on language understanding.

What does this mean for systems? Very simply, pre-processing tools like Named Entity Recognizers are not needed. Databases on named entities are not needed. However, a system still has to identify entities, especially the characters in a narrative text, as they remain major actors throughout the whole text, and no prior knowledge on them is available.

6.1.2.2 Question

Questions are of two types. Here are some examples:

(46) Who helped create and spread the modern image of butterflies?

(47) What does the writer say about butterflies in this passage?

(48) Many people are now uneasy about increased leisure because

(49) The main point the writer wishes to make is that

Either the question is really a full question, with a "Wh" interrogative and a question mark, or it is the beginning of a sentence ended by each answer choice. Full questions are almost never of a factoid nature, like our first example. A quick sampling of the Wh-questions indicate that only 10 to 12% of them are of a factoid nature, and they seem even more difficult that the other questions, because it requires the system to gather the necessary entities from the text, with no other

indication than the question. Special handling of those questions seems pointless as a first approach.

The other type of question, the incomplete sentence, varies in usefulness from essential to useless: in the third example, the answer is the reason of the stated fact so it seems important to use it when searching the text; in the fourth example, the answer choices most likely contain the full information and the question is only a mere introduction.

We also note that the questions can refer to "the writer" or "the passage", words that do not designate anything in the textual content, but which could be called "meta-entities". These mentions appear quite frequently, so it seems useful to have some special handling method for them.

From a human reader standpoint, questions feel useful to locate the relevant passage in the text quickly. They rarely feel essential in choosing the correct answer choice however. Once the relevant passage is located, it is often enough to check the passage and each answer choice for coherency. In conclusion, answer choices are really the most important elements in this task.

6.1.2.3 Answer choice

The answer choice is the most interesting component of a reading comprehension test. It is designed to lure weak test takers toward a wrong answer, and still be completely unambiguous, so that strong test takers consistently succeed in finding the right answer. The Entrance Exams are no different, as they were originally meant for human test takers.

Let us look at the full example given in Section 6.1.1 (p. 148). The four answer choices are clearly on topic: they all deal with one of the characters doing climbing. The 3 wrong answers have something interesting and very important in common: all their words, without exception, are present in the text, in the exact same surface form. Even when they are not in the passage we provided, the words can be found somewhere further in the text. What about the right answer choice? Some of its words are actually not included anywhere in the text, like "one of the most difficult". This means that the usually most trusted features as described in the literature review, like word overlap and other surface form matching methods, are actually not to be trusted at all, or at least not alone.

We ran a test to confirm it: we computed the word overlap of answer choice with the text, eliminated the answer choice with the highest score among them, and picked at random one of the three remaining answer choices. This way, we obtained an accuracy of 0.27, better than picking at random one of the four choices,

which yields 0.25. It also means that an elementary "reasonable" baseline method will perform worse than a nonsensical method like what we just described. We conclude that a system *needs* to capture some form of lexical-semantic variation or sentence structure to be able to simply pick the right answer more frequently than an incorrect answer.

In our submissions, we thought about using this knowledge and meta-reasoning about multiple-choice question structure to improve our performance, for example by eliminating the closest answer for surface matching. This however is not helpful in the grand scheme of things, which is to solve a general textual inference problem, not a textual inference problem purposely rigged to trick people. In our three years of participation to this task, we never came across any submission which used this kind of assumptions.

Observing the answer choices, we also realize that finding the relevant passage is actually a difficult task in itself. This confirms that reading comprehension tests are a really complete form of CQI. The task most likely requires to implement all capabilities described in Chapter 3, including searching the context (the short texts of the test) through the query (the questions), with corresponding rule:

$$\frac{Q?K \vdash T \longrightarrow H}{Q?K \cup K' \vdash T \longrightarrow H}\ \text{ctxt-search}$$

If we plan to apply passage retrieval on the text, we need to be able to evaluate this step independently. This is why we describe the annotation of the corpus in passages in the next section.

6.1.3 Corpus annotation

The Entrance Exams dataset is provided with the gold standard answers. These annotations are easy to do anyway, because it simply comes down to answering the reading test like a test taker would. We need another layer of annotation on passages.

A *passage* is any kind of sequence of contiguous sentences from the text, usually between one and four. Then we can define the *relevant passage* for questions, correct answer choice, incorrect answer choice or any combination of those elements – but mostly, a question plus an answer choice. The relevant passage is the passage in the text where one of these notions is expressed in its closest meaning. The passage is often unique and rather obvious: due to the short size of the text, a competent human reader can most of the time find unambiguously what the

question/answer choice refers to in the text. More rarely, the question or answer choice is not expressed anywhere in the text, or at least at no one location that we can pinpoint. This happens on approximately one question in twelve for the correct answer, but more frequently for the incorrect answer choice. Even more rarely, we need two different passages in the text that are far apart to answer the question: this happened 3 times in the entire combined corpus for the 3 years. So the existence of a passage containing something very close in meaning to what the question or answer choice is about is a rather safe assumption.

We manually annotated the relevant passages for question and individual answer choices, with the BRAT annotation tool (Stenetorp et al., 2012). These annotations are far from requiring an important expressive power of the annotating rules, they are mostly continuous spans of words in the text. Annotators are still invited to be as precise as possible when annotating a passage, possibly skipping over irrelevant elements. This does not mean the passages are actually disjoint, just that the final complete passages may contain extra information. Different human annotators may include more context sentences than others, but where the passages are centered around is otherwise pretty much an easy consensus, especially for question plus answer choice combined. Two answer choices to the same question can share a relevant passage when they are very similar in meaning.

Let us consider an example. The following excerpt is the beginning of a text in the 2014 dataset.

(50) "We're going," Mimi called out to her mother in the family's grocery store next to her house. This was her first date, and Robert Rovere had just arrived to take her to a dance. She could hardly believe it was happening. During the long wait she had wondered again and again what to wear, finally putting on her favorite blouse. Now at last Robert was here. He looked beautiful to her. His hair was neatly combed and he wore a yellow sweater she hadn't seen before. Mimi felt wonderful.

The question is :

(51) What particular point suggests that Mimi was nervous about her date?

We annotate as relevant passage for this question the segments: "This was her first date" and "During the long wait she had wondered again and again what to wear, finally putting on her favorite blouse". These are sufficient as they contain the topic of the question: Mimi's date and Mimi's nervousness. Let us look at the answer choices.

The answer choices are:

1. She could hardly believe she had taken such a long time to get ready.
2. She kept Robert waiting for a long time until she was ready.
3. She spent a long time making herself look nicer. (correct answer)
4. She wondered many times whether Robert would like the yellow sweater.

Our annotations are respectively:

1. She could hardly believe it was happening
2. Robert Rovere had just arrived to take her to a dance
3. During the long wait she had wondered again and again what to wear, finally putting on her favorite blouse
4. His hair was neatly combed and he wore a yellow sweater she hadn't seen before.

In the 1st answer choice, "she could hardly believe" is targeted as the main topic. The 2nd passage is the passage which directly contradicts the 2nd answer choice while dealing with the same topic. The 3rd passage corresponds to the correct answer, and justifies it nicely. The 4th passage is the only mention of a yellow sweater anywhere in the text, and this is clearly the topic of the answer choice, so this is again an easy annotation.

In the process of making these annotations to allow further evaluation of a passage retrieval step, we gained valuable insight on how to solve the textual entailment step once we assume we can reliably retrieve the relevant passage. A hybrid method of validation and invalidation might be successful. Indeed, on the previous example, it is easy to invalidate the 1st answer choice: its first part and its second part are completely unrelated in the text. But it is not easy to validate the 3rd answer choice because its formulation is quite distant from what is expressed in the relevant passage.

When studying the invalidating passages, we found three main causes of invalidity. In a second annotation phase, we only labeled incorrect answer choice with these causes. The three possible labels are:

- missing
- context

- fact

Missing is the idea that the answer choice is not expressed in the text: a relevant passage is hard to find. This generally happens for the following reasons: the only possible passage is too long (more than 5 sentences); words of the answer choice are all found in the text, but never linked as in the answer choice; the answer choice simply introduces a foreign concept never expressed anywhere in the text, or only through a single polysemous word. *Context* indicates that the context – the passage – indeed contains the answer choice, but the answer choice still does not answer the question. We figure it is important in this case to rely on context to make this conclusion. *Fact* represents the most common cause, where the answer choice is just not factual in the text. The related information is present in the document, often easy to find because the answer choice uses the same words as the passage, but it is at the very least incorrect, and sometimes even contradictory. It can come from a lot of mismatches: subject/argument inversions, polarity/modality, antonyms, temporality ...

In the next section, we try to leverage these types of error cause identified during the annotation process in our approach to validation and invalidation as a decision method for this task.

6.2 Validation and invalidation

The validation of an answer choice requires a system to do two things: one is confirming that the relevant passage is consistent with the answer and even justifies it, the other is checking that the answer choice does address the question – it cannot be any random fact contained in the text.

Invalidating an answer choice is obviously doing the opposite, but one must only find one incoherent element between the answer choice and its passage: one argument inversion is enough, one polarity mismatch is enough.

We try to use this duality to our advantage in our submissions to the Entrance Exams task. First, we do not opt for a machine learning method as we deem the training data not big enough in size (in 2014), so we design manual validation and invalidation rules that turn out to be ineffective on the test data. Then, we design and implement a machine learning method, hoping that this time we have enough data (in 2015). This method was successful and obtains the second best result at the most recent evaluation.

6.2.1 Manual rules

Our manual validation and invalidation method is used in our submission at CLEF 2014. This work was not particularly successful and most of its components are already described in this thesis, so we will keep the description of the system short. A complete description can be found in the working note published at CLEF 2014 (Gleize et al., 2014).

6.2.1.1 System overview

The system starts by ranking passages by their relevance to the question, using the same method as described in Section 4.2.1 (p. dbpr). It then extracts from the passages and the answer choices predicate-argument structures (PAS) built from pruned dependency parsing. A procedure aligns the predicates and the arguments in the passage and the answer choice, with a 1-to-1 similarity measure based again on our structured lexical expansion and an Integer Linear Programming optimization. We summarize the information available in those alignments. For each aligned word pair, we know:

- the function that each word plays in its respective PAS: subject, predicate or argument.
- a semantic relatedness score.
- truth value annotations, as annotated by *TruthTeller*.

TruthTeller (Lotan et al., 2013) is a semantic annotator that assigns truth values to predicate occurrences in a sentene. More details about the annotation types are available in its presentation paper, but we generally only use the Predicate Truth annotation, which is the final value assigned by TruthTeller. It is one of P (Positive), U (Uncertain) or N (Negative), and indicates whether the predicate itself is entailed by its containing sentence, in the classical sense of textual entailment.

6.2.1.2 Decision by validation/invalidation

Our goal in the last step of our system is to eliminate as many answer choices as possible without eliminating the right answer. If it eliminates too many answer choices or not enough, our system chooses not to answer (which is quite unusual when compared to the other submission at CLEF). Let K be the maximum number of answer choices we are allowed to keep to take the final decision

(in experiments, $K = 2$). The following algorithm computes whether we take a final decision for the current question. The `align` function returns PAS alignments (PASAlignments). The `validate` and `invalidate` functions apply our manual rules.

```
ALL_ANSWERS := all answer choices
AnswersChoices := all answers
Passages := all relevant passages
While (|AnswersChoices| > K && |Passages| > 0) {
  Passage = Passages.pop()
  Validated = {}
  Invalidated = {}
  Foreach (AnswerChoice in AnswerChoices) {
    PASAlignments = align(AnswerChoice, Passage)
    Foreach (PASAlignment in PASAlignments) {
      if (validate(PASAlignment))
        Validated.add(AnswerChoice)
      if (invalidate(PASAlignment))
        Invalidated.add(AnswerChoice)
    }
  }
  ToRemove := {}
  if (|Invalidated| < |ALL_ANSWERS|) {
    ToRemove := Invalidated
  if (|Validated| > 0) {
    ToRemove := ToRemove U (ALL_ANSWERS \ Validated)
  AnswersChoices := AnswersChoices \ ToRemove
}
return |AnswersChoices| <= K
```

We provide a rough explanation of what goes on in this pre-decision process. We explore all the ranked relevant passages as long as we have not eliminated enough answers. When faced with a new passage, we align the PAS of each answer choice with the passage. The ranked alignments go through validation and invalidation rules and the corresponding answer is invalidated if a PAS alignment is found invalid.

Finally, we present our rules. We manually built 2 validation rules and 3 invalidation rules. They operate on PAS alignments. The validation rules must be fired simultaneously to validate an answer choice, whereas only one invalidation

rule fired is enough to invalidate an answer choice. This corresponds to what we wrote earlier about the relative difficulty of validating, compared to invalidating. In the description of the rules, *polarity* means the predicate truth value of the aligned words. Compatible polarities are P with P, N with N, and U with P,N,U. Strong alignment means that the word-to-word alignment score is above a threshold, manually set in this system.

The validation rules are as follows:

- **Rule 1**: Subject and Predicate are strongly aligned in both PAS, and all polarities are compatible.
- **Rule 2**: Predicate and one Argument are strongly aligned in both PAS, and all polarities are compatible.

The invalidation rules are as follows:

- **Rule 1**: One polarity mismatch is found in a strong alignment.
- **Rule 2**: Predicates are strongly aligned, but their Subjects are not aligned at all.
- **Rule 3**: The alignment is located at least 2 sentences before the best (question, passage) alignment. We noticed on the trial corpus that the correct answer was usually found after the mention of the question in the document.

These rules are fairly intuitive, except maybe for the third invalidation rule. It checks that the answer choice is expressed after the question is expressed in the text. An order inversion usually means that answer choice and question are unrelated.

6.2.1.3 Results

Trial refers to the Entrance exams dataset of 2013 while *Test* refers to the 2014 version. Table 6.1 reports global results of our system on both sets of questions. As we can see, performance is quite satisfactory on the trial dataset, but really poor –at the level of the random baseline in c@1– on the test dataset.

We also analyze in a fine-grained way the accuracy and efficiency of our validation and invalidation rules. Table 6.2 describes the error rate on questions where invalidation rules end up eliminating a correct answer choice. In each cell, we find

	Trial	Test
Questions answered	42/60	36/60
Errors	22	25
Accuracy	0.48	0.31
c@1	0.43	0.25

Table 6.1: Results on trial and test

	Trial	Test
Rule 1	2 / 21	4 / 13
Rule 2	6 / 25	8 / 22
Rule 3	8 / 26	10 / 28

Table 6.2: Invalidation errors on trial and test

the number of questions where the particular rule eliminated a correct answer, and we also find the number of questions it fires on –where it eliminates at least one answer choice. As we can see, all rules misfire more on the test dataset, which seems to correlate well with the overall poor test performance. Validation rules do not fire as often as the invalidation rules (4 times for trial, and 5 times for test).

It is possible to frame our rule system as a kind of information retrieval system and evaluate it in term of validation precision and recall, and invalidation precision and recall. For validation, relevant items are the correct answers, and for invalidation, relevant items are the incorrect answer choices. We compare our results in both trial and test datasets with a random baseline, which basically has a 50% chance of validating and invalidating each answer choice and stops under the same conditions as our system (when K or less answer choices remain). Results in term of precision and recall are shown in table 6.3. As we can see, surprisingly, validation/invalidation do not seem to perform significantly differently from random guesses on test data, while it is not the case at all on trial data.

Barring a bug in our implementation, we do not have a satisfying explanation for this, except that we may have over-tuned our rules on trial data. The test data also presents format differences with the trial data, but surely not enough to affect the results that much. A lot of the modules in our system can be responsible for the performance loss on test data. However, one could not reasonably expect impressive results with only those few manual validation and invalidation rules. Our next – and last – contribution presents a validation and invalidation machine-learning method based on tree edit beam search.

System	Random	Rules (Trial)	Rules (Test)
Validation precision	0.24	**0.36**	0.25
Validation recall	0.40	**0.57**	0.42
Invalidation precision	0.75	**0.83**	0.74
Invalidation recall	0.60	**0.66**	0.56

Table 6.3: Precision/recall of validation/invalidation

6.2.2 Learning from Tree Edit Beam Search

This last contribution to the Entrance Exams task refers to the machine learning method described in Section 5.2 (p. 135). We already described the core of the method, the Tree Edit Beam Search algorithm. The full architecture is described in Figure 6.1. Its pipeline is composed of mainly five modules: preprocessing, passage retrieval, graph enrichment, beam search with tree edit model and final classifiers for validation/invalidation. The graph enrichment module refers to the addition of ConceptNet relations which we also already talked about. In this section, we mostly focus on the passage retrieval and validation/invalidation modules, as well as experiments and error analysis.

6.2.2.1 Passage retrieval

The passage retrieval module aims at extracting relevant short snippets from the document to pass to the more computationally expensive modules further down the pipeline. Words of the question and the answer choice act as the query of CQI. However, it is very rare that words of the question exactly appear in the relevant passage of the document, so we have to use some form of query expansion.

We enrich the lemmas with coreference information, WordNet relations (synonyms, antonyms, hypernyms, hyponyms), and weigh the words by the IDF score of the original word in the document.

If the words of the query are not found using the previous expansion methods, we use a vector-based representation of words to compute a similarity measure. Word vectors are those provided by Huang et al. (2012). To each word, we assign a vector of 50 values. Huang et al.'s resource actually provides multiple vectors for each word, to account more accurately for polysemy, so we use the same window-based disambiguation method as the author to compute the right one. We then pair the query word vectors with the document word vectors with the highest cosine similarity. We also take into account bigram vectors, by summing

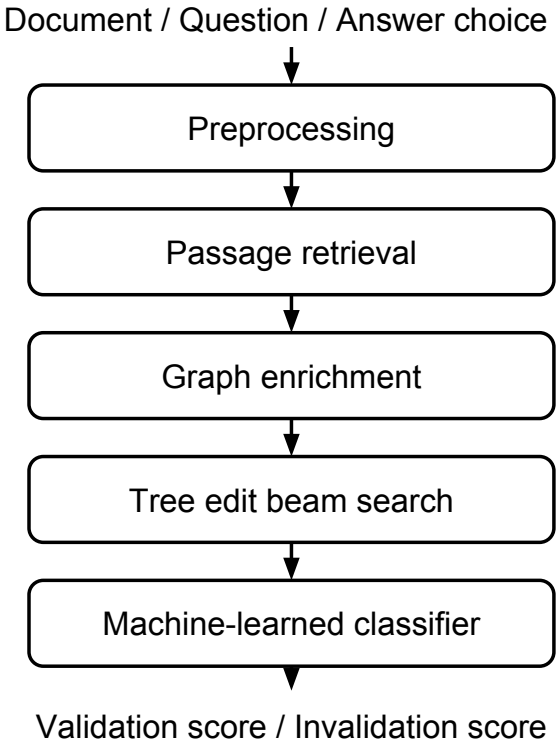

Figure 6.1: System Architecture at CLEF 2015

2 vectors, which means that we can effectively handle 1-to-2, 2-to-1 and 2-to-2 word scored alignments.
Passages are ranked according to the scoring function defined by Equation 6.2 and are then naturally extended to the full sequence of sentences they span.

$$score(passage) = \frac{\#matchedWords}{\#queryWords} \times \sum_{i=1}^{n-1} \frac{score(w_i) + score(w_{i+1})}{dist(i, i+1)^2} \quad (6.2)$$

We take into account the potential absence of query words by multiplying the passage score by the fraction of query words the passage contains. Each document word $w_i \in \{w_1, ..., w_n\}$ matching a query word is given a simple alignment $score$ (1 if they have same lemmas, 0.9 if they are WordNet synonyms, 0.8 if they are in another WordNet relation, and their word vector cosine similarity otherwise), weighted with the IDF of the word, and the formula is normalized by the square of the distance between the words in the sentence.
This passage retrieval method retrieves a lot of short passages, most of which will overlap or will not be correct, but the Tree Edit Beam Search which uses them is designed to handle numerous source passages.

6.2.2.2 Training validation and invalidation classifiers

The classifier pair, for validation and invalidation, uses the feature set defined in Section 5.2.3.3 (p. 141). We experimented with two models, logistic regression and random forest, both implemented in Weka (Hall et al., 2009), and results are presented in the next subsection. It must be clear that we use only one of those two models to learn both classifiers with the same feature set: the features characterize an edit sequence, and an edit sequence can transform a passage into an incorrect answer choice, or turn it into a correct choice. The only thing we change between the two classifiers is their training set. We focus here on how we built the training data.

What we want to avoid is trying to learn how to transform any random text snippet in the document into any random answer choice, because it serves no purpose. Indeed, as readers, we cannot validate the right answer choice by looking at a couple of arbitrary sentences in the text, nor can we invalidate a wrong answer choice if the passage we are reading is not even related to the question. Thus, during the training phase, only our annotated relevant passages are selected (cf Section 6.1.3), and the algorithm runs on them, without a passage retrieval phase.

We create the learning (passage, answer choice) pairs by annotating them following the semantics described in Figure 6.2. In this figure, RP stands for right

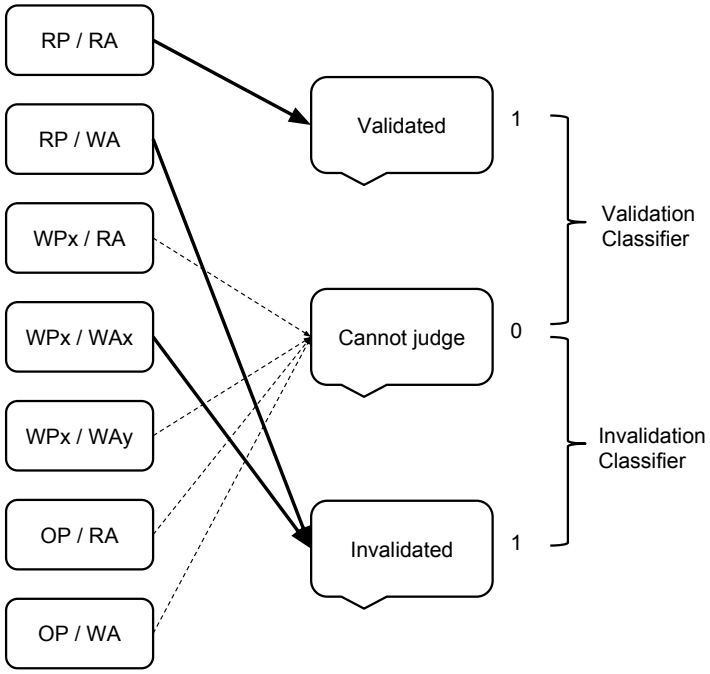

Figure 6.2: Semantics of (passage, answer) pairs

passage, RA for right answer, WA for any wrong answer, WAx for the wrong answer choice x, WPx for the passage expressing it, OP for any other passage than the one expressing the paired answer choice. To summarize, the only time we can either validate or invalidate are when we operate on passages relevant to some answer choice: we annotate as validated only if we have both the right passage and the right answer, and invalidated if we have a wrong answer choice with either the passage which expresses it in the document or the right passage. This follows the intuition that as readers, given a question, a passage and an answer choice, we can probably tell if the provided passage is self-sufficient in expressing the right answer to the question or if there is a mismatch between an answer choice and the passage in the text it refers to.

Then the edit sequences for this data are computed, their features are extracted, and sequences for both classifiers are labeled using the aforementioned semantics. Implicitly, as this is not visible in Figure 6.2, if an edit sequence is labeled 1 (valide/invalidate) for one classifier, it is labeled 0 for the other. The thin dashed arrows simply symbolize that the label is 0 for both classifiers.

	Random forest	Logistic regression
Questions answered	89	89
Errors	57	61
Accuracy	0.360	0.314
# of tests with c@1 ≥ 0.5	8	4
c@1	0.360	0.314

Table 6.4: Our results on Entrance exams at CLEF 2015

System	c@1
Synapse	**0.58**
LIMSI (our system)	0.36
cicnlp	0.30
NTUNLG	0.29
CoMiC	0.29

Table 6.5: Best results of all teams at CLEF 2015

For the test run, the algorithm runs on the test data, and the answer is chosen based on the regression numbers output by the two classifiers. First, for each answer choice, the edit sequence with the highest *max*(`validationScore`, `invalidationScore`) is selected. Ideally we want an edit sequence which is characteristic of either a high confidence validation, or a high confidence invalidation, so that we may classify the answer choice confidently as either correct or incorrect in the next step. Then, the answer choice whose selected sequence has the highest `validationScore` $-$ `invalidationScore` is finally picked: we want as much separation as possible between the validation and invalidation scores of correctly classified answer choices as possible. If they have a similar validation and invalidation score, the system behaves as if randomly guessing between validation and invalidation.

6.2.2.3 Results

Table 6.4 is a simple reminder of our results, already described in Section 5.2.3.3 (p. 141). Table 6.5 lists the best results of all teams which submitted runs to the evaluation. The next section deals with error analysis.

6.2.2.4 Error analysis

A pertinent qualitative analysis is always delicate to do for machine learning systems with such low performances. It is indeed always possible to draw examples that look like the system is obviously supposed to correctly handle but end up as errors. Conversely, it is always possible to find a complex instance on which the system somewhat miraculously worked (i.e. made a lucky guess).

Nevertheless, we first report some of the simple errors that our system made. In the following passage/question pair, our system got lured by answer 3, closest in surface form to the relevant passage. ConceptNet does not link "held" to "trapped", and "its original nature" from the correct answer could not be linked to anything in the passage (it is however found further in the text).

Several years ago, certain scientists developed a way of investigating the nature of the atmosphere of the past by studying air caught in the ice around the North or South Pole. According to their theory, when snow falls, air is trapped between the snowflakes. The snow turns to ice with the air still inside.
Certain scientists claimed that 1) atmospheric gases increase the yearly amount of snow 2) falling snowflakes change the chemical balance of the air 3) the action of atmospheric gases causes snow to turn into ice **4)** the air held between snowflakes keeps its original nature (correct)

In the following passage/question pair, our system picked the answer choice 3. It would have been easy to pick the correct answer 1 if "wrong" could have been linked to "mistake", but in ConceptNet, this is a *RelatedTo* relation, which we did not consider. We realize that there is actually a lot of information in those *RelatedTo* relations, and ideally our system should handle them, but we decided in the design phase to remove them because they are not semantically precise.

Everyone stared. That was embarrassing enough, but it was worse when I finished my coffee and got ready to leave. My face went red - as red as his hair - when I realized I'd made a mistake.
The woman's face turned red **1)** because she realized that she had been quite wrong about the boy (correct) 2) because she realized that the boy was poor and hungry 3) because she saw everyone staring at her 4) because she hated being shouted at

In both those cases, a more precise characterization of correct passages would have been useful, because in the first case, our answer choice skips over the sentence which contains the correct answer, and in the second case, the sentence containing our answer choice appears way before the sentence containing both question and correct answer.

Finally we report an example of correctly answered question through mostly invalidation. In the following passage/question pair, our system easily invalidated answer choice 1 (due to the added negation) and answer choice 4 (due to the first sentence of the passage saying the opposite). Then, answer choice 2 had edit sequences which hinted at both validation and invalidation, so it was still a risky pick (but with slightly more invalidation). In the end, the remaining answer choice (3), for which the system found neither validation nor invalidation, was correctly picked by default.

Kate was an energetic woman who expected people always to be doing something, and she found plenty of jobs for Fred to do. This made him feel part of the household, but now he really wanted to be able to sit and reflect on the events of his life. If he had continued to live alone, he would have had the time to do this to his heart's content. One afternoon he felt he simply had to get away from the house. "I'm going for a walk," he said, closing the door behind him. Leaving the town, he walked across the fields and followed a slow-moving stream toward the hills. After a while he came to a pool in the stream under some trees. Here, he thought, was a place he could come to when he needed to reflect on the past. Although the stream seemed unlikely to have any fish, he would simply tell Kate he had found a place to go fishing. When he mentioned the stream that night, his son-in-law, Jim, said in disbelief, "There aren't any fish there. That stream runs dry half the summer."
Why did Fred tell Kate that he had found a place to go fishing? 1) He didn't feel part of the household with Kate and Jim. 2) He enjoyed fishing very much and was glad to be able to do it again. **3)** He wanted a way to leave the house without hurting Kate's feelings. (correct) 4) He was bored in the house because there were few things to do.

As for quantitative analysis, the general trend is that our system performs better when edit sequences remain short, with over 40% accuracy when the chosen edit sequences are shorter than 6 edits (on average on all the answer choices). We considered this was still not significant enough of an advantage to choose not to answer questions based on a length threshold of edit sequences to improve our results on c@1.

6.2.2.5 Discussion

In this part, we present an original method of training two classifiers using the same features and learning algorithms but asymmetric training sets. This allows us to implement a pair of distinct validation and invalidation classifiers. This has the obvious benefit of leveraging our passage annotations on the whole Entrance Exams corpus, instead of having to determine validation/invalidation rules by hand, like in our first system.

As of now, we do not have a clear idea of the impact of each classifier separately. We would like to address it in the future.

Also, we did not take advantage of the possibility to choose not to answer a question. In our experience, every missed answer adds variance when running on the test set: we are evaluated on even fewer questions, and the initial dataset is already quite small. Thus, we did not prioritize exploiting this feature of the evaluation. However, we believe our learning method has the potential to handle it. In future works, it would be interesting to design a meta-classifier working on the output of the two current classifiers.

6.3 Conclusion

This chapter concludes our thesis by tackling a complete Contextually Queried Inference problem: reading comprehension tests. Our study and annotations of the corpus built throughout three years of Entrance Exams tasks at CLEF leads to methods based on validation and invalidation. The first one suffers from not having enough training data and being constrained to manual rules probably overfitting the trial set. The second one builds on better performing elements of this thesis and instances a lot of capabilities of CQI. It uses a passage retrieval method including a lot of resources and metrics to pinpoint the most relevant context to the query. Its tree edit model captures sentence rewriting which is a simplified form of inference chaining. Its decision functions actually solve a textual entailment problem more than a question answering problem. The use of relations in concept graphs like ConceptNet could point to identifying lacks of information: for example, a tree node enriched by the relation "HasPrerequisite" begs the question of said prerequisite in the text: what is it? Is it fulfilled?

The duality of validation and invalidation of a textual inference did not appear in Chapter 3 when we formulated the capabilities we deem essential to solve CQI. One might wonder if these two processes are only two sides of the same coin, or really two essential notions we have to integrate in a CQI problem, as some sort of "coherency checks". When we designed CQI, we preferred viewing validation and invalidation as a choice of implementation to solve the problem rather than being part of its definition.

Conclusion

As Natural Language Processing heads toward Natural Language Understanding – and in a way goes back to it, as NLU was always one of the main goals in AI –, textual inference becomes an increasingly important goal. It is currently evaluated in multiple ways, across various tasks and various datasets. While the diversity is welcome because it allows to tackle multiple linguistic phenomena, this lack of unity can also be detrimental: the research community is divided and the greater goal may drift out of sight. This thesis first attempted to reconcile several different tasks around the core features and algorithms they rely on. We proposed a hierarchical taxonomy which defines ways of classifying inference tasks in term of their difficulty, regardless of their evaluation format. We then defined a more generic problem, *Contextually Queried Inference* (CQI), encompassing most inference-related tasks. A formal system implements the theoretical capabilities we deem essential for its resolution. We view these two contributions, the taxonomy and CQI, as complementary. The first focuses more on the pratical tools and resources required to solve the problems, and the second spawns a theoretical discussion about inferential capabilities.

In this thesis, we also implemented methods to solve concrete aspects of textual inference. All our methods either solve the full CQI problem – at least the task closest to it – or are easily transferable to a wide range of CQI-related applications. Our *structured lexical expansion* uses a simplification hypothesis to solve both passage retrieval and textual entailment. The idea is that the simpler the lexicon, the easier it is to compare the sense of two sentences. The results on passage retrieval are not conclusive, most likely due to the static nature of our manually-defined similarity measures. However, the results on textual entailment are promising, which might be due to the availability of training data at the evaluation campaign we took part in.

After these systems, we came to the conclusion that we needed a machine learning model capable of seamlessly integrating structural similarity and lexical-

semantic resources. Such a model would have to be robust and sufficiently expressive. We proposed a *type-enriched string rewriting kernel* which fits those requirements perfectly. The ability to incorporate various resources to loosen the word-to-word matching is the originality of our contribution, and it allows our system to be competitive on multiple tasks: paraphrase identification, textual entailment and question answering. We showed that the kernel still runs efficiently compared to the untyped kernel, but it still suffers from limitations in scalability: the running time of kernel-based machine learning methods like SVM depends quadratically on the number of training examples.

From the beginning, the main application of this thesis was meant to be reading comprehension tests. They are an easy way to test machine reading and deep textual inference capabilities. We frame their multiple-choice questions as a dual task of validation and invalidation: sometimes, it is easier to invalidate an incorrect answer choice than validate the correct one. After early experiments with manual rules which allowed us to better study the problem, we implemented two classifiers extracting their features from a tree edit model called *Tree Edit Beam Search*. This algorithm shares a lot with the string rewriting kernel: uses rewriting rules, includes lexical-semantic resources, captures syntactical transformations. It also has the benefit of having a low dimension feature set, as well as scaling naturally well with the number of training examples: a logistic regression only depends linearly on it. Our corpus annotation of passages on CLEF 2013-2015's data provided us with two asymmetric training sets, which is how we produced two classifiers for validation and invalidation. With this system, we obtained the second best result at the Entrance Exams task at CLEF 2015. This is a satisfying result, but systems remain closer to random guessing than to the performance of competent human readers. Looking at the top systems, it is clear that the current methods will eventually reach a performance ceiling on this task, if it has not happened already. We think that general textual inference will remain unsolved in the near future and that we will ultimately need something more, which makes it all the more interesting to work on.

Future directions

In this section we describe interesting topics which would build on our work and extend it. We first present short-term goals, which represent the direct continuation of this work. Then we close this thesis with long-term goals, which are what we think our research field could investigate in the next few years.

Our short-term goals focus mainly on our two most important contributions, type-enriched string rewriting kernels (TESRK) and tree edit beam search (TEBS). Our kernel method is very promising, but cannot scale in its current version to large datasets. We already had troubles running the method on a personal computer, and needed 64-core computers to comfortably test different sets of types on datasets not usually considered sizable, like RTE 1-2-3. What we could investigate is scalable adaptations for kernel methods, like the one of Dai et al. (2014). These methods usually rely on stochastic heuristics to reduce the dimension of the problem. Succeeding in implementing a scalable kernel method would allow to train models on huge datasets, like the very recent large corpus for textual inference from Stanford (Bowman et al., 2015), containing 570,000 inference pairs (for comparison, RTE 3 contains 800 inference pairs).

Currently, TESRK only handles types on single words, which are only useful when matching one word with another. It would be interesting to see if the model can be modified to include types on several words, in order to include, for example, background paraphrase knowledge. A quick idea is to encode a type on a span of words with a set of types on single words. The single types would encode that they are preceded or followed by another specific single type, part of the same multi-word relation (like the pattern of a paraphrase rule). Adding paraphrase rules using this method would have a very positive impact on the expressive power of the model: currently, TESRK cannot model a true chaining of several inference rules, like our CQI formulation advocates: rewriting rules are just applied directly from one existing sentence to another. Extra paraphrase rules as types would model that we can apply an extra intermediary rewriting rule inserted in the middle of this process.

A number of experiments and additions can be attempted on TEBS and the full system used at CLEF 2015. This is the last system we were able to implement but we lacked the time to do multiple interesting experiments. First, in the current iteration, the scores of validation and invalidation are combined using a simple manually-defined formula. It would be easy to combine them with an additional meta-classifier. A much deeper analysis of the model, its parameters and the results is required to really judge the impact of each feature and hyper-parameter. What is the impact of the beam width, the number of top passages considered, or the number of edit sequences extracted? How does each individual classifier perform on a subtask of validation or invalidation? These are all questions that have to be answered before deciding to make any change to the system.

Regarding our structured lexical expansion, especially the successive application of paraphrase rules to solve textual entailment, we still think there is some-

thing to be done with dictionary definitions used as highly structured paraphrases. The Simple English Wiktionary definitely lacks in coverage, but the standard English Wiktionary is one of the largest lexical resources available today. More than 800,000 definition sentences could make for a good paraphrase database if parsed and transformed correctly.

The most urgent long-term goal in textual inference is surely acquiring more training data. We think the future of textual inference data is not as bright as one might think, compared to factoid question answering and machine translation. The current problem is that textual inference does not translate directly into a commercial need or user experience. People do not ask complicated questions to their phone (maybe because they cannot be answered today, but probably more because they have less of a need for immediate answers to tough questions). Finding the answer may require some kind of inference to fill missing entities, but overall, easy factoid questions predominate on smart search engines: the top 10 questions asked from Google in 2011 were all "What is" definition questions. Thus, there is not currently a high interest in manually producing a large dataset for textual inference, nor is it easy to produce automatically. Bowman et al. (2015)'s massive new inference dataset is certainly promising, but has appeared too recently for us to really have an idea of the positive impact it might bring. This depends a lot on the types of inference featured by their inference pairs.

Paradoxically, there exists a vast untapped wealth of reading comprehension tests. The problem is that these tests are actually great commercial assets, so the datasets are generally not freely available. Indeed, each year a lot of reading comprehension tests are produced, for use in standardized testing of students throughout the world. However, the tests are re-used after the exam as part of books of practice tests sold to the next wave of students.

We did not address IBM's Watson in this thesis (Ferrucci et al., 2010). Watson was first and foremost designed as a powerful factoid question answering engine, and we know that it leverages plenty of resources and machine learning methods, but it is actually hard to evaluate its true inferential capabilities. As Watson's popularity continually grows, the shroud of mystery around its inner workings seemingly thickens. Not much can confirm it, but Watson might already have very strong textual inference capabilities.

In the end, perfect inference capabilities may rest on the advent of a *strong AI*, an artificial intelligence functionally equal to that of humans. We believe this is unlikely to be achieved on pure linguistic data, like texts. As mentioned in our hierarchy of inferences, the hardest problems may require other senses available to humans, like sight, taste, smell or touch.

Bibliography

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics, 2012. 26

Elif Aktolga, James Allan, and David A Smith. Passage reranking for question answering using syntactic structures and answer types. In *Advances in Information Retrieval*, pages 617–628. Springer, 2011. 62

Giuseppe Attardi, Luca Atzori, and Maria Simi. Index expansion for machine reading and question answering. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012. 99

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007. 78

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998. 72

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4, 2006. 56

Philip Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1):217–239, 2005. 64

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009. 77

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008. 78

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015. 172, 173

Chris Brockett. Aligning the rte 2006 corpus. *Microsoft Research*, 2007. 57

Chris Brockett and William B Dolan. Support vector machines for paraphrase identification and corpus construction. In *Proceedings of the 3rd International Workshop on Paraphrasing*, pages 1–8, 2005. 26, 57, 59

Fan Bu, Hang Li, and Xiaoyan Zhu. String re-writing kernel. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 449–458. Association for Computational Linguistics, 2012. 53, 81, 117, 118, 121, 127, 129, 131, 134

Kate Cain, Jane V Oakhill, Marcia A Barnes, and Peter E Bryant. Comprehension skill, inference-making ability, and their relation to knowledge. *Memory & cognition*, 29(6):850–859, 2001. 15

Hiram Calvo, Andrea Segura-Olivares, and Alejandro García. Dependency vs. constituent based syntactic n-grams in text similarity measures for paraphrase recognition. *Computación y Sistemas*, 18(3):517–554, 2014. 117

Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004. URL `https://jena.apache.org/`. 72

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3): 27, 2011. 127

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. Discriminative learning over constrained latent representations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 429–437. Association for Computational Linguistics, 2010. 70, 134

Nick Chater, Joshua B Tenenbaum, and Alan Yuille. Probabilistic models of cognition: Conceptual foundations. *Trends in cognitive sciences*, 10(7):287–291, 2006. 19

Philipp Cimiano, Vanessa Lopez, Christina Unger, Elena Cabrio, Axel-Cyrille Ngonga Ngomo, and Sebastian Walter. Multilingual question answering over linked data (qald-3): Lab overview. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 321–332. Springer, 2013. 78

Peter Clark, Philip Harrison, and Xuchen Yao. An entailment-based approach to the QA4MRE challenge. In *CLEF (Online Working Notes/Labs/Workshop)*. Citeseer, 2012. 28

Vincent Claveau, Ewa Kijak, and Olivier Ferret. Improving distributional thesauri by exploring the graph of neighbors. In *International Conference on Computational Linguistics, COLING 2014*, pages 12–p, 2014. 59

Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968. 39

Courtney Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics, 2005. 55

Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pages 255–262. Omnipress, 2010. 67

Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics, 2011. 67

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407. ACM, 2005. 62, 69, 81, 103, 104, 105

Peter W Culicover. Paraphrase generation and information retrieval from stored text. *Mechanical Translation and Computational Linguistics*, 11(1-2):78–88, 1968. 106

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer, 2006. 1, 9, 10, 20, 21, 43, 56

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014. 172

Dipanjan Das and Noah A Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics, 2009. 69, 81

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, 2014. 78

Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008. 107

Marie-Catherine De Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D Manning. Learning to distinguish valid textual entailments. In *Second Pascal RTE Challenge Workshop*, 2006. 62

Michael Denkowski and Alon Lavie. Meteor-next and the meteor paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 339–342. Association for Computational Linguistics, 2010. 56

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc., 2002. 50

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004. 50, 56

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*, 2005. 128

Myroslava O Dzikovska, Diana Bental, Johanna D Moore, Natalie B Steinhauser, Gwendolyn E Campbell, Elaine Farrow, and Charles B Callaway. Intelligent tutoring with natural language support in the beetle ii system. In *Sustaining TEL: From Innovation to Learning and Practice*, pages 620–625. Springer, 2010. 106

Myroslava O Dzikovska, Rodney D Nielsen, and Chris Brew. Towards effective tutorial feedback for explanation questions: A dataset and baselines. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 200–210. Association for Computational Linguistics, 2012. 111, 112

Myroslava O Dzikovska, Rodney D Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. SemEval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 263–274. Association for Computational Linguistics, 2013. 32, 44, 51, 106

Katrin Erk and Sebastian Pado. Shalmaneser-a flexible toolbox for semantic role assignment. In *Proceedings of LREC*, volume 6, 2006. 73

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical*

Methods in Natural Language Processing, pages 1535–1545. Association for Computational Linguistics, 2011. 78

Vanessa Wei Feng and Graeme Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, USA, June*, 2014. 75

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010. 31, 173

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013, Beijing, China, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P15-1097`. 67, 81

Dan Flickinger. On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(01):15–28, 2000. 72

Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. Overview of todai robot project and evaluation framework of its nlp-based problem solving. *World History*, 36:36, 2014. 148

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `http://cs.jhu.edu/~ccb/publications/ppdb.pdf`. 78

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007. 17

Martin Gleize and Brigitte Grau. Limsiiles: Basic english substitution for student answer assessment at semeval 2013. In *Second Joint Conference on Lexical and*

*Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 598–602, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S13-2100. 93, 105

Martin Gleize and Brigitte Grau. A hierarchical taxonomy for classifying hardness of inference tasks. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 3034–3040, 2014. URL http://www.lrec-conf.org/proceedings/lrec2014/summaries/1168.html. 28

Martin Gleize and Brigitte Grau. Limsi-cnrs@clef 2015: Tree edit beam search for multiple choice question answering. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*, 2015a. URL http://ceur-ws.org/Vol-1391/115-CR.pdf. 94, 136

Martin Gleize and Brigitte Grau. Noyaux de réécriture de phrases munis de types lexico-sémantiques. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 170–181, Caen, France, June 2015b. Association pour le Traitement Automatique des Langues. URL http://www.atala.org/taln_archives/TALN/TALN-2015/taln-2015-long-015. 94, 117

Martin Gleize and Brigitte Grau. A unified kernel approach for learning typed sentence rewritings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 939–949, 2015c. URL http://aclweb.org/anthology/P/P15/P15-1091.pdf. 94, 117

Martin Gleize, Brigitte Grau, Van-Minh Pho, Anne-Laure Ligozat, Gabriel Illouz, Frédéric Gianetti, and Loïc Lahondes. Selecting answers with structured lexical expansion and discourse relations limsi's participation at QA4MRE 2013. In *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013.*, 2013. URL http://ceur-ws.org/Vol-1179/CLEF2013wn-QA4MRE-GleizeEt2013.pdf. 93, 98

Martin Gleize, Anne-Laure Ligozat, and Brigitte Grau. Limsi-cnrs@clef 2014: Invalidating answers for multiple choice question answering. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, pages 1386–1394, 2014. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-QA-GleizeEt2014.pdf`. 95, 157

Nizar Habash and Bonnie Dorr. Catvar: A database of categorial variations for english. In *Proceedings of the MT Summit*, pages 471–474, 2003. 58

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009. 163

Catherine Havasi, Robert Speer, and Jason Alonso. Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent advances in natural language processing*, pages 27–29. Citeseer, 2007. 78

Michael Heilman and Nitin Madnani. Ets: domain adaptation and stacking for short answer scoring. In *Proceedings of the 2nd joint conference on lexical and computational semantics*, volume 2, pages 275–279, 2013. 52

Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics, 2010. 65, 81, 117, 131, 132, 134, 145

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. Recognizing textual entailment with lcc's groundhog system. In *Proceedings of the Second PASCAL Challenges Workshop*, 2006. 78, 81, 134

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. 60

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012. 161

Hen-Hsen Huang, Kai-Chun Chang, and Hsin-Hsi Chen. Modeling human inference process for textual entailment recognition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 446–450, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P13-2079`. 29

Aminul Islam and Diana Inkpen. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V*, 309:227–236, 2009. 117

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, 2014. 77

Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997. 67

Sergio Jimenez, Claudia Becerra, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. Softcardinality: hierarchical text overlap for student response analysis. In *Proceedings of the 2nd joint conference on lexical and computational semantics*, 2013. 52, 117

PN Johnson-Laird, Sangeet S Khemlani, and Geoffrey P Goodwin. Logic, probability, and human reasoning. *Trends in cognitive sciences*, 19(4):201–214, 2015. 19

Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2000. 73

Panayiota Kendeou, Paul Broek, Anne Helder, and Josefine Karlsson. A cognitive view of reading comprehension: implications for reading difficulties. *Learning Disabilities Research & Practice*, 29(1):10–16, 2014. 15

Paul Kingsbury and Martha Palmer. From treebank to propbank. In *LREC*. Citeseer, 2002. 72

Anne Kispal. Effective teaching of inference skills for reading. Technical report, Research Report DCSF-RR031). Cheshire, UK: National Foundation for Educational Research, Department of Education (Division of Children, School and Families), 2008. 15, 16

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003. 140

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289, 2001. 58, 65, 69

Dominique Laurent, Baptiste Chardon, Sophie Nègre, and Patrick Séguéla. English run of synapse développement at entrance exams 2014. *CLEF 2014 Working notes*, 2014. 74

Dominique Laurent, Baptiste Chardon, Sophie Nègre, Camille Pradel, and Patrick Séguéla. Reading comprehension at entrance exams 2015. *CLEF 2015 Working Notes*, 2015. 74, 144

C Leacock and M Chodorow. Combining local context and wordnet sense similarity for word sense identification. wordnet, an electronic lexical database, 1998. 55

Gary Geunbae Lee, Jungyun Seo, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Changki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, et al. Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *TREC*, 2001. 101, 102

Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1986. 55, 98

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966. 50

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011. 31, 86

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *KR*, 2012. 18, 45

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. 142

Elisabeth Lien and Milen Kouylekov. Semantic parsing for textual entailment. *IWPT 2015*, page 40, 2015. 72

Marc Light, Gideon S Mann, Ellen Riloff, and Eric Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7 (04):325–342, 2001. 102

Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics, 2003. 55

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 768–774. Association for Computational Linguistics, 1998. 59, 72, 109

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360, 2001. 78

Mihai C Lintean and Vasile Rus. Dissimilarity kernels for paraphrase identification. In *FLAIRS Conference*, 2011. 117

Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning toolkit. *BT technology journal*, 22(4):211–226, 2004. 141

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002. 134

Amnon Lotan, Asher Stern, and Ido Dagan. Truthteller: Annotating predicate truth. In *HLT-NAACL*, pages 752–757, 2013. 157

Bill MacCartney and Christopher D Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200. Association for Computational Linguistics, 2007. 19, 29

Bill MacCartney, Michel Galley, and Christopher D Manning. A phrase-based alignment model for natural language inference. In *Proceedings of the conference on empirical methods in natural language processing*, pages 802–811. Association for Computational Linguistics, 2008. 57

Nitin Madnani and Bonnie J Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387, 2010. 106, 118

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics, 2012. 50, 56, 81, 117

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. The excitement open platform for textual inferences. *ACL 2014*, page 43, 2014. 79

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*. CIDR 2015, 2014. 78

William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988. 74

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`. 77, 99

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of com-

positional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*, 2014. 72

Suguru Matsuyoshi, Yusuke Miyao, Tomohide Shibata, Chuan-Jie Lin, Cheng-Wei Shih, Yotaro Watanabe, and Teruko Mitamura. Overview of the ntcir-11 recognizing inference in text and validation (rite-val) task. In *Proceedings of the 11th NTCIR Conference*, pages 223–232, 2014. 148

Gail McKoon and Roger Ratcliff. Inference during reading. *Psychological review*, 99(3):440, 1992. 28

Adam Meyers, Catherine Macleod, Roman Yangarber, Ralph Grishman, Leslie Barrett, and Ruth Reeves. Using nomlex to produce nominalization patterns for information extraction. In *Proceedings: the Computational Treatment of Nominals, Montreal, Canada,(Coling-ACL98 workshop)*, volume 2, 1998. 58

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006. 117

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010. 60

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 60

George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 33, 34, 54, 117

Shachar Mirkin, Roy Bar-Haim, Jonathan Berant, Ido Dagan, Eyal Shnarch, Asher Stern, and Idan Szpektor. Addressing discourse and document structure in the rte search task. *Proc. of TAC*, 2009a. 59

Shachar Mirkin, Ido Dagan, and Eyal Shnarch. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 558–566. Association for Computational Linguistics, 2009b. 60

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003. 29

Robert C Moore. Towards a simple and accurate statistical approach to learning translation relationships among words. In *Proceedings of the workshop on Data-driven methods in machine translation-Volume 14*, pages 1–8. Association for Computational Linguistics, 2001. 59

Thomas Morton, Joern Kottmann, Jason Baldridge, and Gann Bierner. Opennlp: A java-based nlp toolkit. http://opennlp.sourceforge.net, 2005. 77, 127

Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer, 2006. 67, 134, 139

Peter Norvig. Inference in text understanding. In *AAAI*, pages 561–565, 1987. 14

Jane Oakhill. Constructive processes in skilled and less skilled comprehenders' memory for sentences. *British Journal of Psychology*, 73(1):13–20, 1982. 15

Charles Kay Ogden. *Basic English: A general introduction with rules and grammar*. K. Paul, Trench, Trubner, 1944. 97

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002. 50

Anselmo Peñas and Eduard Hovy. Filling knowledge gaps in text for machine reading. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 979–987. Association for Computational Linguistics, 2010. 28, 113

Anselmo Peñas, Eduard H Hovy, Pamela Forner, Álvaro Rodrigo, Richard FE Sutcliffe, Corina Forascu, and Caroline Sporleder. Overview of qa4mre at clef 2011: Question answering for machine reading evaluation. In *CLEF (Notebook Papers/Labs/Workshop)*, pages 1–20. Citeseer, 2011. 23, 45

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, and Roser Morante. QA4MRE 2011-2013: Overview of question answering for

machine reading evaluation. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 303–320. Springer, 2013a. 38

Anselmo Peñas, Yusuke Miyao, P Forner, and N Kando. Overview of qa4mre 2013 entrance exams task. In *CLEF (Online Working Notes/Labs/Workshop)*, pages 1–6, 2013b. 24, 45, 101

Anselmo Peñas, Yusuke Miyao, Álvaro Rodrigo, Eduard Hovy, and Noriko Kando. Overview of clef qa entrance exams task 2014. In *CLEF (Online Working Notes/Labs/Workshop)*. CLEF, 2014. 74

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`. 60

Martin F Porter. Snowball: A language for stemming algorithms, 2001. 127

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*, pages 1–10, 2004. 64, 81

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34 (2):257–287, 2008. 77

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics, 2006. 71, 73, 81

Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge, 2012. 49

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. The life and death of discourse entities: Identifying singleton mentions. In *HLT-NAACL*, pages 627–633, 2013. 140

P Resnik. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995. 55

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2, 2013. 74

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995. 59

Álvaro Rodrigo, Anselmo Peñas, Yusuke Miyao, Eduard Hovy, and Noriko Kando. Overview of clef qa entrance exams task 2015. *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum*, 2015. 142

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. Learning answer-entailing structures for machine comprehension. In *Proceedings of ACL*, 2015. 74, 75, 77

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. 59

Roger C Schank, Neil M Goldman, Charles J Rieger III, and Christopher Riesbeck. Margie: Memory analysis response generation, and inference on english. In *IJCAI*, pages 255–261, 1973. 14

Rolf Schwitter. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1113–1121. Association for Computational Linguistics, 2010. 97

Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *EMNLP*, pages 458–467, 2013. 67, 81

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21, 2007. 72

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231, 2006. 50

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127, 2009. 56

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011. 70, 81, 117

Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2330–2336. AAAI Press, 2011. 58

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972. 59

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012. 154

Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3):422–433, 1979. 63

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM, 2003. 55, 101

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003. 39, 140

Alan M Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950. 31

Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010. 60

Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. 122

Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2000. 118

Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998. 66

Evgenii E Vityaev, Leonid I Perlovsky, Boris Ya Kovalerchuk, and Stanislav O Speransky. Probabilistic dynamic logic of cognition. *Biologically Inspired Cognitive Architectures*, 6:159–168, 2013. 19

Ellen M Voorhees. The trec question answering track. *Natural Language Engineering*, 7(04):361–378, 2001. 23

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006, 2006. 63, 81, 117, 127

Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712, Beijing, China, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P15-2116`. 60, 70

Mengqiu Wang and Christopher D Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172. Association for Computational Linguistics, 2010. 65, 69, 81, 117, 134

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007. 49, 65, 69, 70, 81, 132

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015. URL `http://arxiv.org/abs/1502.05698`. 29, 75

Wiktionary. Wiktionary, the free dictionary. https://en.wiktionary.org/, 2002. 58

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994. 55

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013a. 65, 81, 133

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. A lightweight and high performance monolingual word aligner. In *ACL (2)*, pages 702–707, 2013b. 58

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-markov phrase-based monolingual alignment. In *EMNLP*, pages 590–600, 2013c. 58

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007. 34, 78

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1744–1753, 2013. URL `http://aclweb.org/anthology/P/P13/P13-1171.pdf`. 49, 58, 59, 60, 70, 81, 117

Deniz Yuret, Aydin Han, and Zehra Turgut. Semeval-2010 task 12: Parser evaluation using textual entailments. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 51–56. Association for Computational Linguistics, 2010. 72

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. Shallow semantics in fast textual entailment rule learners. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 72–77. Association for Computational Linguistics, 2007. 66, 81, 117, 134

Fabio Massimo Zanzotto, Lorenzo Dell'Arciprete, and Alessandro Moschitti. Efficient graph kernels for textual entailment recognition. *Fundamenta Informaticae*, 2010. 67, 69, 117, 134

Rolf A Zwaan and Gabriel A Radvansky. Situation models in language comprehension and memory. *Psychological bulletin*, 123(2):162, 1998. 36

Titre : Inférence textuelle pour la compréhension de texte

Mots clés : compréhension automatique de la langue naturelle, inférence textuelle, question réponse

Résumé : Étant donnée la masse toujours croissante de texte publié, la compréhension automatique des langues naturelles est devenue un enjeu majeur de l'intelligence artificielle. En langue naturelle, les faits exprimés dans le texte ne sont pas nécessairement tous explicites : le lecteur humain infère les éléments manquants grâce à diverses compétences que n'ont initialement pas les ordinateurs. Une inférence textuelle est définie comme une relation entre deux fragments de texte : un humain lisant le premier peut raisonnablement inférer que le second est vrai. Beaucoup de tâches de évaluent plus ou moins directement la capacité des systèmes à reconnaître l'inférence textuelle. Au sein de cette multiplicité de l'évaluation, les inférences elles-mêmes présentent une grande variété de types.

Nous nous interrogeons sur les inférences textuelles d'un point de vue théorique et présentons deux contributions répondant à ces niveaux de diversité. La reconnaissance automatique d'inférence textuelle repose aujourd'hui presque toujours sur un modèle d'apprentissage, entraîné à l'usage de traits linguistiques variés sur un jeu d'inférences textuelles étiquetées. Cependant, les données spécifiques aux phénomènes d'inférence complexes ne sont pour le moment pas assez abondantes pour espérer apprendre automatiquement la connaissance du monde et le raisonnement de sens commun nécessaires. Cette thèse propose des systèmes intégrant apprentissage d'alignements entre les mots de phrases et utilisation de connaissances tirées de ressources externes.

Title: Textual Inference for Machine Comprehension

Keywords: natural language understanding, textual inference, question answering

Abstract: With the ever-growing mass of published text, natural language understanding stands as one of the most sought-after goal of artificial intelligence. In natural language, not every fact expressed in the text is explicit: human readers infer what is missing through various skills that computer systems do not initially have. A textual inference is simply defined as a relation between two fragments of text: a human reading the first can reasonably infer that the second is true. A lot of different NLP tasks more or less directly evaluate systems on their ability to recognize textual inference. Among this multiplicity of evaluation frameworks, inferences themselves are not one and the same and also present a wide variety of different types.

We reflect on inferences for NLP from a theoretical standpoint and present two contributions addressing these levels of diversity.

Automatically recognizing textual inference currently almost always involves a machine learning model, trained to use various linguistic features on a labeled dataset of samples of textual inference. However, specific data on complex inference phenomena is not currently abundant enough that systems can directly learn world knowledge and commonsense reasoning. This thesis presents systems which tighltly integrate the learning of word-to-word alignments and the use of external background knowledge.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France